NIUF 403R1-92

North American ISDN Users' Forum Application Software Interface (ASI)

Part 1: Overview and Protocols (Version 1.0)

Approved: October 4, 1991

Updated: October 30, 1992

Application Software Interface Expert Working Group
ISDN Implementors' Workshop
North American ISDN Users' Forum

Revision History

November 1991	Baseline Approved Document (NIU/91-0013)
May 1992	Editorial Corrections (NIU/91-0013)
October 1992	V.120 service (NIUF 403-92)
	Service enhancements

Abstract

This document is a specification for an Application Software Interface (ASI) as defined by the Application Software Interface Expert Group of the North American ISDN Users' Forum.

This document is Part 1 of the ASI. It provides an initial specification intended to allow implementors to begin using the ASI for implementations of applications requiring a limited subset of ISDN services within a limited set of operating systems. The specification includes the following components:

- introduction to the ASI concepts,
- description of the ASI architecture,
- description of the ASI access functionality,
- ASI command messages to support a basic subset of ISDN services,
- and ASI data structures.

Keywords

application programming interface; API; Application Software Interface; ASI; implementation agreement; integrated services digital network; ISDN

Notice of Disclaimer

This specification was developed and initially approved by organizations participating in the North American ISDN Users' Forum (NIUF) meetings in October 1991. Additions have been approved at subsequent meetings. The National Institute of Standards and Technology (NIST) makes no representation or warranty, express or implied with respect to the sufficiency, accuracy, or use of any information or opinion contained herein. The use of this information or opinion is at the risk of the user. Under no circumstances shall NIST be liable for any damage or injury incurred by any person arising out of the sufficiency, accuracy, or use of any information or opinion contained herein.

Acknowledgments

NIST would like to acknowledge the NIU-Forum Application Software Interface Expert Working Group, and especially the following individuals, for their valuable contributions to this document:

Kenneth A. Argo	Fujitsu Networks Industry, Inc.
Reginald Best	Teleos Communications, Inc.
Ron Bhanukitsiri	Digital Equipment Corp.
Cheng T. Chen	Teleos Communications, Inc.
Howard Cunningham	UDS Motorola
Stephen Halpern	NYNEX Science and Technology
Frank Heath	Rockwell CMC
Dory Leifer	Univ. of Michigan
Tauras Liubinskis	Trilan Systems
Jim Loehndorf	Ameritech Services
Chris Nix	IBM
David Rife	Hayes Microcomputer Products
Mike Robins	AT&T Bell Labs
Stephen Rogers	Electronic Data Systems
Chris Schmandt	MIT Media Lab
Ben Stoltz	Sun Microsystems, Inc.
Robert E. Toense	NIST
Adrian Viego	Bellcore
Wayne Yamamoto	Sun Microsystems, Inc.

1.0. Introduction To The ASI

1.1. Overview

This document is a specification for an Application Software Interface (ASI) as defined by the Application Software Interface Expert Group of the North American ISDN Users' Forum.

This document is Part 1 of the ASI. It provides an initial specification intended to allow implementors to begin using the ASI for implementations of applications requiring a limited subset of ISDN services within a limited set of operating systems. The specification includes the following components:

- introduction to the ASI concepts,
- description of the ASI architecture,
- description of the ASI access functionality,
- ASI command messages to support a basic subset of ISDN services,
- and ASI data structures.

Future parts of the ASI specification will expand the above list to include:

- ASI access method for DOS, UNIX/POSIX, OS/2, Windows, etc.;
- additional ASI command messages to support additional ISDN services;
- formal specification of the ASI;
- expansion to include the full teleservices architecture;
- and conformance tests.

1.2. Charter

The Application Software Interface Group is one of the expert working groups within the North American ISDN Users' Forum; ISDN Implementors' Workshop (IIW).

The Application Software Interface focuses on the definition of a common application interface for accessing and administering ISDN services provided by hardware commonly referred to in the vendor community as Network Adaptors (NAs) and responds to the applications requirements generated by the ISDN Users' Workshop (IUW).

The characteristics of this Application Interface shall be:

- portable across the broadest range of system architectures;
- extensible;
- abstracted beyond ISDN to facilitate interworking;
- and defined in terms of services and facilities consistent with OSI layer interface standards.

1.3. Goals

The primary goal of the ASI is to provide a consistent set of application software interface services and application software interface implementation agreement(s) in order that an ISDN application may operate across a broad range of ISDN vendor products and platforms.

The application software interface implementation agreements will be referenced by (and tested against) the IUW generated applications. It is anticipated that the vendor companies involved in the development of these implementation agreements will build products for the ISDN user marketplace which conform to them.

ASI Implementation Agreements are likely to become a U.S. Government Federal Information Processing Standard (FIPS).

ASI specifications are expected to serve as a contribution for further North American or International standards activities.

1.4. Purpose

Today there exists an ever increasing number of ISDN Network Adaptors (NAs) from different manufacturers, each with the same basic subset of features, plus additional features the manufacturers hope will differentiate them from the competition. This environment is illustrated in Figure 1.1.

Currently, each NA vendor presents a different software interface to the ISDN application. This produces constant frustration to the ISDN applications developer. Each interface represents the efforts on the part of the vendor to provide access to all ISDN services provided by the NA; yet each, done in isolation, differs from the others. In developing an ISDN application, therefore, the developer is faced with the task of (a) binding with an initial NA, and (b) once his application is fielded, working to enhance his application product to interface with other NAs as well. Exemplifying this process are products in the market today which advertise “currently works with Network Adaptors A, B, and C; will support Network Adaptors D and E in the near future”.

.

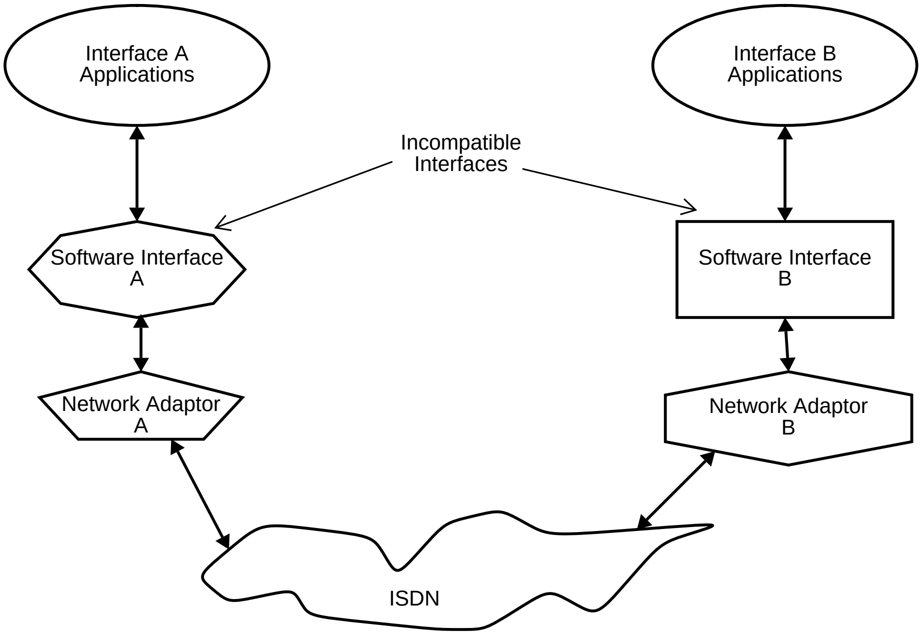


Figure 1.1 - Typical Proprietary Interface Environment.

1.5. Scope

Figure 1.2 conceptualizes a solution to the application interface incompatibility problem.

The ASI is a software interface between an application and a NA within an operating environment (the operating environment includes the operating system, hardware platform, bus, etc.). Elements on the network side of the interface are referred to as the **“ASI Entity (AE).”** Elements on the application side are referred to as the **“Program Entity (PE).”**

The ASI does not guarantee interoperability between the end to end applications that may use the ASI.

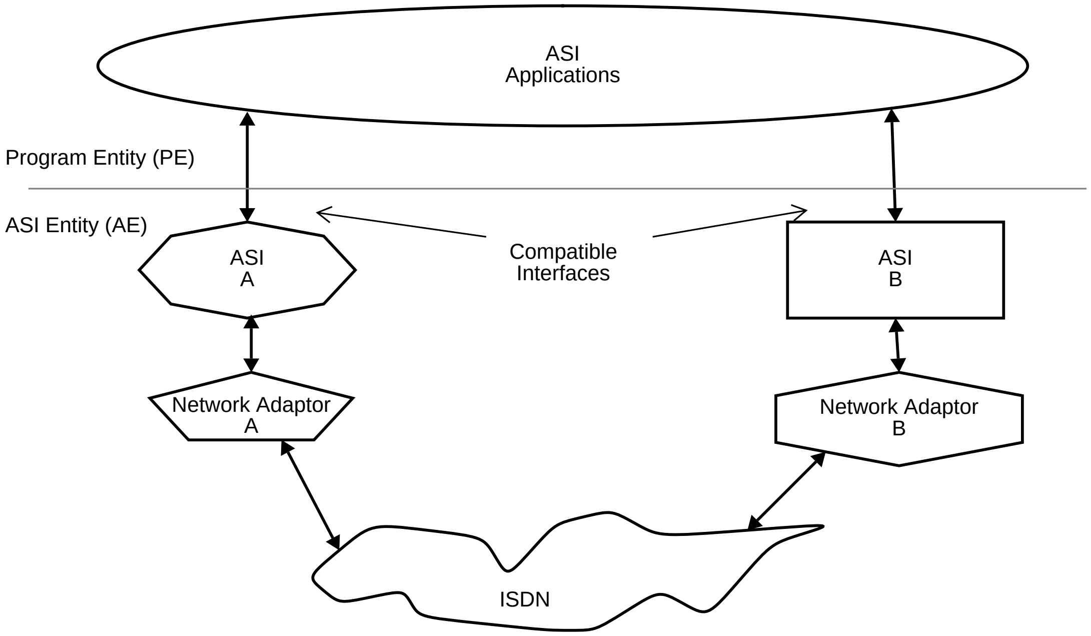


Figure 1.2 - ASI Environment.

The ASI places emphasis on a common application interface as opposed to a common hardware device interface for two main reasons:

1. The most important user benefit is derived from a large selection of commercially available ISDN applications which can operate over a correspondingly large selection of NAs. The number of applications will be most influenced by the existence of a common application interface that allows the application provider to easily migrate applications to different NAs or operating system environments.

2. It is much more difficult to specify a standard hardware device interface. Vendors want to provide different NA hardware interfaces to appeal to different markets. For example, some NAs will be built for performance while others will be built for low cost. The market that a vendor desires to sell into will determine the hardware device interface (i.e., memory mapped, polled I/O, interrupt driven, direct memory access (DMA) driven, shared memory, etc.). Vendors are accustomed to providing drivers or libraries which interface to their specific hardware implementation.

The conversion from the common ASI to the NA hardware device interface becomes the job of the adaptor developer. The conversion function can, for instance, reside in a device driver which is provided by the adaptor manufacturer. The application developer should have to do as little as possible to port an application written for one operating system to a different operating system (e.g., to re-compile or re-link is perceived as minimal effort). Also, within one operating system, the application developer should be able to design applications independently of the NA (i.e., the application should work the same and without modification on the variety of NAs available), assuming the NA provides equivalent services.

The conceptual objective of the ASI is to be as independent as possible of:

- Hardware Platform,
- Operating System,
- Data Protocol Type,
- Programming Language,
- Compiler.

Although the ASI takes the approach of developing a common set of services which are applied across a broad range of environments, the access methods are environment dependent. This is true because of hardware restrictions within different operating environments, performance issues, and fundamental operating system differences.

As applicable ISDN standards evolve it is expected that the ASI will evolve to accommodate those applicable standards.

1.6. Assumptions

Several assumptions have gone into the development of the current ASI specification. These assumptions are described as follows:

- ISDN primary rate and basic rate access are assumed to be the network interface to the NA. This does not preclude application of this interface to NAs which interface to other ISDN access methods.
- The ASI provides a uniform software interface defined between the NA and the application. Throughout the ASI specification, the term “ASI entity” is used to refer to the ISDN service provider, and any associated hardware, network adaptor card, or terminal equipment, while the term “Program Entity” is used to refer to the application which uses the ISDN service.
- This specification does not address peer-to-peer protocol or interoperability issues.
- The ASI interface is assumed to be at the OSI layer three to layer four boundary.
- ANSI Standards, NIUF Agreements, and CCITT Recommendataions are the basis for this ASI specification.
- No default values for parameters are assumed by the interface. All parameter values necessary for a message must be supplied in the applicable data structures.

2.0. Technical Overview

This chapter presents an overview of the ASI architecture and the motivation underlying the chosen approach.

The goal of the ASI is to provide a portable, extensive, and layered software interface to ISDN hardware, call control, and services. Portability allows applications to be developed independent of any particular vendor's ISDN offering, and hence ties the success of the application to the penetration of ISDN rather than the future of a single vendor. Portability favors the application developer by making the application available for a wider audience. But widespread application availability will make it easier to use ISDN services, hastening deployment of ISDN lines, and thus ultimately benefitting the hardware (or ISDN-capable computer platform) vendors as well.

It is the intent of the ASI that applications written against the ASI specification should run on one computer platform employing ISDN interface hardware from different vendors without recompilation or linking. The same application should be portable to a different computer platform (with the same operating system) by recompiling, with no changes required to the source code. For a different operating system, there may be some code changes to accommodate differences in the access method.

A problem with designing application software interfaces to ISDN teleservices is the range of level of functionality such an interface could support. A high level interface would provide generic telephony interface functions, while a lower level would more closely match ISDN-specific message and event types. The ASI favors a layered approach, based on experience with the OSI model and numerous examples in distributed computing.

As such, the ASI incorporates a model with several reference points. A multi-tasking operating system will enable multiple processes to gain access to ISDN services through a server architecture which will provide a high level functional interface and event filtering to minimize ISDN-specific knowledge required of the application. This server, or the single application for a server-less or single threaded operating system, in turn communicates with ISDN call control over a lower level interface which more closely mirrors the ISDN protocol. The various reference points will be illustrated later in this chapter.

The current release of the ASI specification defines the core subset of the lower level reference point. It is to this reference point which vendors must supply ASI support, and, once written, early applications can be developed immediately. No vendor-specific software need operate above this reference point, although a vendor may choose to provide higher level support for added value to an ISDN product.

The ASI defines a reference point and a message protocol across that reference point. ISDN call control and hardware specific interfaces will operate below the ASI and be provided by specific vendors. Vendors may also supply an application library, in some specific programming language, to compose messages in the ASI format.

The ASI specifies a complete interface composed of an operating system *dependent* access method, an operating system *independent* message set, and an operating system *independent* message encoding method. An operating system dependent access method allows the rest of the ASI to exist independent of the OS.

Because the message set and encoding method are identical between the various implementations of the ASI for different operating systems, application portability is greatly simplified.

The ASI message set and operating system specific access methods provide an asynchronous interface to ISDN call control. The application makes requests through the ASI, and the ISDN call control beneath the ASI transmits confirmation messages and event indication messages back through the ASI as appropriate. Any blocking or synchronous interface to the ASI should be provided as a library of function calls on the application side of the ASI.

For example, an application places a call by sending an Nb-CONNECT request. After issuing the request, the application can continue execution. Call control may generate various Nb-EVENT indication messages as the call proceeds through the network. When the call completes to the called party, an Nb-CONNECT confirmation message will be sent up through the ASI. To implement a blocking call request, the application would send the Nb-CONNECT request, and await the Nb-CONNECT confirmation.

2.1. Application Software Interface Definition

The Application Software Interface (ASI) is a common interface for accessing ISDN services provided by ISDN network adaptors (NAs). The ASI is a way for an application and an ASI entity to communicate within an operating environment (the operating environment includes the operating system, hardware platform, bus, etc.). The translation of the ASI message set, to and from the instructions needed to operate any hardware interfaces, is accomplished by AE vendor supplied software. The conversion function, can, for instance, reside in a device driver provided with the AE.

The application developer should be able to design applications independent of the NA with which it might be used. Within a given operating environment (e.g., a PC running DOS), applications should be able to run on any ASI-compliant AE. Finally, the application developer should have to do as little as possible (e.g., recompile/relink) in order to move from one operating system to another. The ASI allows any ISDN application written against the ASI specification to communicate with any ASI-compliant ISDN network service provider.

2.2. OSI Reference Model Positioning

The ASI is positioned at the Service Access Point (SAP) between layers 3 and 4 in the OSI Reference Model. Conceptually, the ASI is an asynchronous message stream between the ISDN network services provider (layers 1 - 3) and the user (layers 4+) of those services.

If, for example, a non-empty transport layer protocol is positioned above the ASI, then that transport protocol, and not the higher layer application, is the actual user of the ISDN bearer services provided through the ASI. Likewise, the term ‘ASI entity’ is meant to apply to any provider of ISDN network services that meets certain qualifying assumptions. The ASI is a local interface between layer 4 and layer 3 only; it is not, itself, a layer within the OSI Reference Model, nor is it an end-to-end protocol. Such features as interoperability or end-to-end integrity must be provided by protocols above the ASI, using ISDN network services accessed through the ASI.

2.3. Teleservices Architecture

The environment in which the ASI is expected to operate assumes an architecture including a generic teleservices server. Such a server may offer teleservices to applications on the local machine or on the local area network without requiring the applications to implement the details of ISDN, POTS/PSTN, or other possible teleservices media.

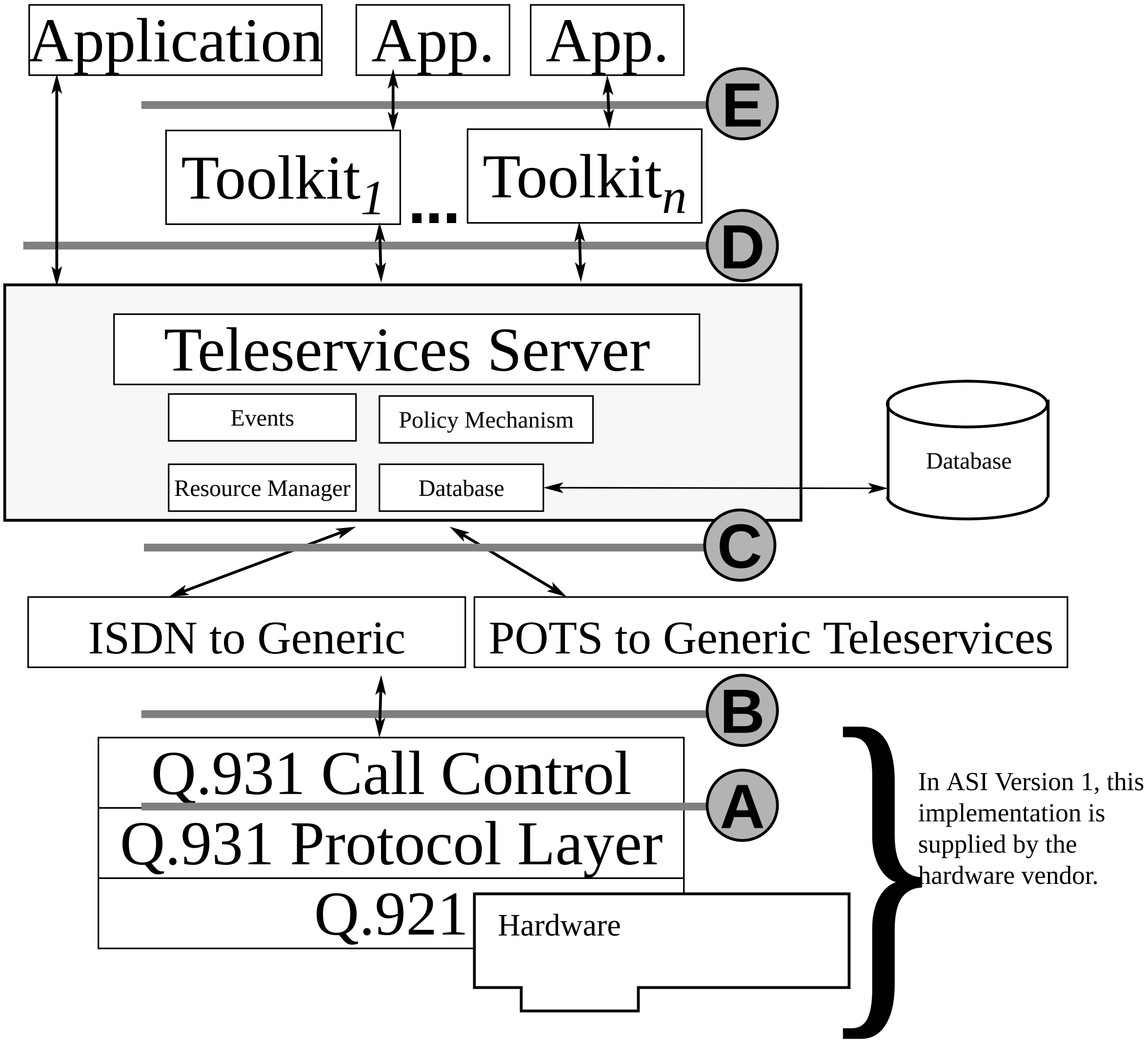
It would be the responsibility of a server interface definition to allow for multiple client applications to access the services provided by a single interface adaptor.

The teleservices architecture has been split into several layers. These layers add functionality to the ASI. They do not imply a progression up the OSI Reference Model. Version 1 ASI is identified as the message stream at reference point ‘B’ in this architecture.

The definition of the Reference Points is as follows:

- In ASI Version 1, the “A” reference point is not an exposed interface. It is defined to be the interface between the “standard” portions of Q.931 and the non-standard portions. Only the non-standard portions of ISDN need to be customized for each market.
- The “B” reference point is the interface between the ISDN signaling, management and user planes, and a server or dedicated application. Direct multi-client access is not allowed.
- The “C” reference point presents a generic teleservices interface to the server or dedicated application.
- The “D” reference point allows a server to provide a generic teleservices interface to multiple client applications. This interface also presents a simplified programming model to the application or toolkit developer.
- The “E” reference point is the programming interface provided by a high level library. This interface is the one most desired by typical applications developers.

Teleservices Architecture for Call Control



2.4. ASI Sessions

The ASI Entity (AE) and Program Entity (PE) communicate across the ASI by reference to sessions. A session is a local virtual path between the PE and the AE which carries all requests and responses for a given instance of a service, e.g., a voice call. Once established, a session is referred to by a session ID.

Sessions are created dynamically by either the AE or the PE according to the rules defined by the ASI protocol. To allow for dynamic creation of sessions by either side, each side may create session IDs without consulting the other side. The AE's session ID is referred to as the AEI, while the PE's session ID is referred to as the PEI. Either side may refer to a session using the other's ID. An ID of all zeros indicates that the other side's ID is unknown or is not used.

Retiring and reuse of old IDs is carefully managed by the protocol.

2.5. ASI Components

The ASI, or any other interface in the architecture, must contain definitions for the following:

- Access methods for each operating environment (DOS, Unix, etc.), for passing messages;

- A set of message types and associated parameters;
- Precisely defined encodings for the above messages;
- A formal description of the protocol semantics.

2.5.1. Access Methods

An access method, as defined by this document, is an operating system dependent set of procedures for passing messages between layers of software. The messages may contain control, management, or user plane information.

This architecture requires that any access method provides asynchronous message passing between software layers.

Access methods are described in Chapter 4.

2.5.2. Messages

In order to meet the portability and network transparency requirements of the architecture, all messages are required to be self contained. Messages containing pointers, or other references, to external data structures are not legal.

Messages and their semantics are described in Chapter 5. Message parameters are described in Chapter 6.

2.5.3. Encoding

Message definition will be described using ASN.1.

Actual message encoding will be done using an ASI specific method. The method is chosen to promote ease of implementation and improve performance while providing for future expansion of the protocol.

3.0. Access Method Functionality

3.1. Overview

3.1.1. Scope

This section provides a philosophy for the Application Software Interface (ASI) Access Methods.

3.1.2. Access Method Philosophy

Although the ASI includes access method definitions specific to each operating system environment (i.e., DOS, UNIX, OS/2, etc.), each of these access methods adheres to a consistent philosophy. That philosophy produces an access method which is optimal for its operating system environment, but results in a consistent software structure across all operating system environments. The application of this philosophy results in the ability to apply a system dependent wrapper around an operating system independent core ASI software module or Program Entity. Additionally, each access method is designed to support:

- The existence and concurrent operation of multiple network adaptors (for further study).
- The existence and concurrent operation of multiple ASI Entities (for further study).
- The existence and concurrent operation of multiple Program Entities (for further study).
- Any combination of multiple/single network, ASI Entities, and/or Program Entities (for further study).
- The concurrent operation of the ASI and other types of adaptors such as Local Area Network (LAN) adaptors (for further study).
- A language independent implementation.
- The highest level of performance possible.
- Minimal use of system resources (memory, soft interrupts, etc.).
- Simple system administration.
- A binary compatible interface (requires no linkage or recompilation).
- A bidirectional asynchronous operation across the interface.
- The dynamic allocation of resources within the ASI Entity.
- A simplified implementation of the Program Entity’s ASI interface.

3.1.3. Access Method Services

Any access method provides mechanisms to transfer information (data, commands, events, etc.) across the ASI. Since the implementation of these mechanisms are operating system dependent, the access method is operating system dependent. The information transfer is independent of the data content.

The access method provides three services common to all operating system environments and consistent with the ISDN protocol reference model. These services are the management, control, and user planes.

The management plane service supports the exchange of all information associated with operation, administration, and maintenance (configuration data, provisioning data, etc.) of the interface and modules that support the interface.

The control plane service provides the ability to control network connections, allocate and deallocate shared resources, change service characteristics, and provide supplementary services.

Use of the user plane is dictated by a connection’s bearer service. These services will be based on the ISDN protocol reference model defined in CCITT Recommendation I.320.

4.0. Access Methods

Access methods for various environments are found in separate parts of this publication.

5.0. Commands

This chapter defines the commands used between the Program Entity (hereafter referred to as **PE**) and the ASI Entity (hereafter referred to as **AE**). At the **B** reference point, ASI commands are actually messages which are exchanged between the **PE** and **AE** and instruct the appropriate entity about events which are taking place, or requests for service. The ASI commands fall into four categories: User -> Network, Network -> User, User -> AE, and AE -> User.

The User -> Network and User -> AE categories are comprised of Requests and Responses. R*equests* are commands which initiate an action. For example, the Nb-CONNECT request is used to place a call. Thus the local terminal is initiating the action. R*esponses* are used to perform actions resulting from commands which were either initiated by the network or distant terminal.

The Network -> User and AE -> User categories are comprised of Indications and Confirmations. I*ndications* are commands which result from actions by the network or distant terminal. I*ndications* either inform the **PE** of events or carry requests for actions by the **PE**. C*onfirmations* are usually messages which conclude a sequence of events which the **PE** initiated via a *request*.

5.1. Terminology

Throughout this chapter, there is certain terminology which is used to describe various ISDN or terminal components. This specific terminology is described below.

Term	Description
local terminal	The CPE which is located at the near end. We assume that there is a known CPE which is connected locally to the network and is issuing or receiving the messages described in this chapter
distant terminal	The CPE at the distant end. This terminal receives the calls *from* the local terminal, and initiates calls *to* the local terminal.
session	For the purposes of describing a call and its associated information, we use the term session. A session is created by an Nb-CONNECT request or an Nb-CONNECT indication. A session is said to be *OPEN* from the moment of its creation until it is terminated by an Nb-DISCONNECT response or an Nb-DISCONNECT confirmation.

Table 5-1: Terminology

5.2. Message Format

ASI commands are passed between the PE and AE using messages which have a fixed header and variable length data. The fixed header is comprised of five fields: Protocol Identifier (**PI**), ASI Entity Identifier (**AEI**), Program Entity Identifier (**PEI**), Command (**CMD**), and Length (**LEN**). The fixed length portion of the header is followed by variable length data. When a command does not require additional parameters, the LEN field is set to ZERO. **NOTE: All multiple octet fields in both the fixed header and additional parameters are in High -> Low order**.

Octet	Contents
1	Protocol Identifier (**PI**)
2	ASI Entity Identifier (**AEI**) - MSB
3	AEI - LSB
4	Program Entity Identifier (**PEI**) - MSB
5	PEI - LSB
6	Command (**CMD**) - MSB
7	CMD - LSB
8	Length (**LEN**) - MSB
9	LEN - LSB
10	Additional Parameters (**AP**)
	:
	:
	:

Table 5-2: Message Format

MSB = Most Significant Byte

LSB = Least Significant Byte

PI The Protocol Identifier is a one octet field arranged as two four-bit nibbles. The high order nibble is used to denote which **interface** is being used (i.e., ASI or other). The low order nibble specifies which **plane** the command is for: Command, Management, or User.

Bits							
8	7	6	5	4	3	2	1
Interface				Plane			

Currently three **Interface** values have been assigned values as shown in Table 3.

Table 5-3: Interface values

Bit	8	7	6	5	4	3	2	1
reserved	0	0	0	0	xx	xx	xx	xx
ASI	0	0	0	1	xx	xx	xx	xx
PCI	0	0	1	0	xx	xx	xx	xx

xx = Don’t care

The four values for the **Plane** are coded as shown in Table 4.

Table 5-4: Plane values

Bit	8	7	6	5	4	3	2	1
reserved	xx	xx	xx	xx	0	0	0	0
Control	xx	xx	xx	xx	0	0	1	0
Management	xx	xx	xx	xx	0	1	0	0
User	xx	xx	xx	xx	0	1	1	0

xx = Don’t care

AEI The **A**SI **E**ntity **I**dentifier is a two octet field which contains an integer. The **AEI** is a unique identifier assigned by the **AE** to identify an *open* session. A value containing all 1's (0xFFFF) is used to specify **ANY** when appropriate. All 0's are used to indicate **DON'T CARE**.

PEI The **P**rogram **E**ntity **I**dentifier is a two octet field which contains an integer. The **PEI** is a unique identifier assigned by the **PE** to identify an *open* session. A value containing all 1's (0xFFFF) is used to specify **ANY** when appropriate. All 0's are used to indicate **DON'T CARE**.

CMD The Command is a two octet field which contains the messages described in this chapter.

LEN A two octet field containing the length of the additional parameters in octets.

AP The Additional Parameter(s) is a variable length data block which is used to pass the ASI data structure elements of the various messages. All additional parameters are coded in ASN.1 and appear in Chapter 6.

5.3. How to Use this Chapter

This chapter is divided into functional components: Call Control, Maintenance and Administration, Supplementary Services, Feature Access, and Miscellaneous Services. In each section, the messages are paired into groups: Requests and Responses, and Indications and Confirmations.

To conserve space and for ease of readability, the standard parameters as described in section 5.2 (AEI, PEI, etc.) are not included with the description of each message. Only those additional parameters which are necessary for each message are included.

5.3.1. Identifying the Interface

After a **PE** has bound to the **AE** using the appropriate access method, it may be necessary in some cases for the **PE** to know which interface the **AE** can respond to (ASI or other). For this purpose, the following special purpose message is defined.

Command: One octet from the PE to AE

PI = 0xFF

Response: Three octets from the AE to PE

PI = 0xFF
Protocol
Version/Revision

Protocol: A one octet field. Each bit position is significant and the following values are defined:

Table 5-5: Protocol values

Bit	8	7	6	5	4	3	2	1
ASI	xx	xx	xx	xx	xx	xx	xx	1
PCI	xx	xx	xx	xx	xx	xx	1	xx

xx = Don’t care

Version/Revision: a one octets field divided into two four bit nibbles

Bits							
8	7	6	5	4	3	2	1
Version				**Revision**			

5.4. Control Plane Functions

5.4.1. Basic Call Control

DESCRIPTION

Basic Call Control services are those required by the **P**rogram **E**ntity(**PE**) to manage the establishment, progress and termination of calls. Establishment refers both to placing outgoing calls and responding to incoming calls.

The names selected for the commands and their parameters are intended to make the discussion as clear as possible and are not intended to be rigorous or derived from standardized terminology.

Also, detailed parameter definitions are not included. For complete details on any parameter, see the appropriate reference in Chapter 6.

DEFINITIONS

The ASI uses an asynchronous interface between the **PE** and **AE**. This means that messages are passed across the interface via the access method in either direction and at any time without waiting for a response. The access method will provide a mechanism to indicate the success or failure of the inter-entity transmission only. Error conditions which result from the processing of a message in either the **AE** or **PE** will be reported to the other entity via the appropriate message.

In the ASI, a session can be initiated by either the **AE** or **PE**. When the AE initiates the session, then the AEI alone is sufficient to identify the session in all subsequent messages. However, when the **PE** initiates the call, it would be difficult to co-ordinate tracking which messages belonged to the call because it would have no corresponding reference in the messages from the **AE**. For this reason, all Call Control messages have an **AEI-PEI** pair. This allows the **PE** to assign a value which the **AE** will echo in all future messages associated with the open session. Until the **PE** establishes which **AEI** the **AE** assigned to the outgoing call, it can track the messages via the **PEI**.

5.4.1.1. Commands

This section contains the definitions and descriptions of the messages which are sent from the **PE** to the **AE**. R*equests* initiate actions (i.e., place calls) independent of other stimuli. R*esponses* generate actions as a result of *indications* from the **AE**. I*ndications* are messages which result from incoming Q.931 messages. C*onfirmations* are messages which conclude a previous request.

5.4.1.1.1. **Nb-CONNECT** confirmation [**0x0101]**

DESCRIPTION

Nb-CONNECT confirmation confirms the successful completion of a call to the distant terminal and includes a completed session block.

ADDITIONAL PARAMETERS

session block A block of data which contains the mandatory and/or optional data elements required by this message. The following types of sessions are currently defined by the ASI:

- voice call
- provisioned PSD (B or D channel)
- Circuit Switched Data (CSD), Clear channel (no rate adaption)
- CSD, V.120, call owner
- CSD, V.120, call user
- CSD, X.25 end-to-end or to a PSN, call owner
- CSD, X.25 end-to-end or to a PSN, call user
- I.515 negotiated rate adaption (for further study)
- V.110 (for further study)

5.4.1.1.2. **Nb-CONNECT** indication [**0x0102]**

DESCRIPTION

Nb-CONNECT indication is an indication of a new call which has arrived and is waiting for the local terminal to accept or reject the call.

ADDITIONAL PARAMETERS

session block A block of data which contains the mandatory and/or optional data elements required by this message. The following types of sessions are currently defined by the ASI:

- voice call
- provisioned PSD (B or D channel)
- Circuit Switched Data (CSD), Clear channel (no rate adaption)
- CSD, V.120, call owner
- CSD, V.120, call user
- CSD, X.25 end-to-end or to a PSN, call owner
- CSD, X.25 end-to-end or to a PSN, call user
- I.515 negotiated rate adaption (for further study)
- V.110 (for further study)

Comments

During the setup phase of a call, the session block may not contain all of the possible parameters. The Nb-CONNECT confirmation will contain a completed session block. The Nb-CONNECT indication will contain a valid AEI and a PEI of DON'T CARE.

5.4.1.1.3. **Nb-CONNECT** request [**0x0103]**

DESCRIPTION

Nb-CONNECT request is used to place both voice and data calls. Since various types of calls (i.e., Voice or CSD) have multiple modes of operation, all mandatory parameters must be included as well as those optional parameters which are required for the particular call type and mode. See Chapter 6 for a complete parameter list for each session type.

ADDITIONAL PARAMETERS

session block — A block of data which contains the mandatory and/or optional data elements required by this message. The following types of sessions are currently defined by the ASI:

- voice call
- provisioned PSD (B or D channel)
- Circuit Switched Data (CSD), Clear channel (no rate adaption)
- CSD, V.120, call owner
- CSD, V.120, call user
- CSD, X.25 end-to-end or to a PSN, call owner
- CSD, X.25 end-to-end or to a PSN, call user
- I.515 negotiated rate adaption (for further study)
- V.110 (for further study)

COMMENTS

Nb-CONNECT request results in a SETUP message being sent to the NETWORK. For this message, the PEI is significant and the AEI should be set to DON'T CARE. A valid AEI will be assigned by the **AE** and returned with the next *indication* or *confirmation*.

5.4.1.1.4. **Nb-CONNECT** response [**0x0104]**

DESCRIPTION

Nb-CONNECT response is used to accept an incoming call and is generated as a result an Nb-CONNECT indication message which is pending. While this response does not require any additional parameters, the **PE** may include optional parameters such as 7kHz audio in order to establish a negotiation sequence.

ADDITIONAL PARAMETERS

Required:	None
Optional:	Any parameter from the CSV session block

5.4.1.1.5. **Nb-CONNECT_STATUS** confirmation [**0x0105]**

DESCRIPTION

This confirmation is the return from an Nb-CONNECT_STATUS request.

ADDITIONAL PARAMETERS

Channel	Channel being used by this session
Associations	IDs of other sessions associated with this session
LCN	Logical Channel Number
Owner/User	Parent/Child relationship indicator for current session and its associated session(s)
Call state	Status of connection (i.e., alerting, held, etc.)

5.4.1.1.6. **Nb-CONNECT_STATUS** request [**0x0106]**

DESCRIPTION

This request is used to find out the status of an ISDN connection. A value of ANY in the AEI and DON'T CARE in the PEI will cause the AE to send one Nb-CONNECT_STATUS confirmation per valid AEI. A valid AEI-PEI pair will result in a single Nb-CONNECT_STATUS confirmation being returned.

ADDITIONAL PARAMETERS

Required:	None

5.4.1.1.7. **Nb-DISCONNECT** confirmation [**0x0131]**

DESCRIPTION

Nb-DISCONNECT confirmation advises the PE that the distant terminal has responded to its Nb-DISCONNECT request. Upon receiving this confirmation, the PE discards any information it is holding for the current PEI-AEI pair and marks the PEI as available for reuse.

ADDITIONAL PARAMETERS

Required:	None

5.4.1.1.8. **Nb-DISCONNECT** indication [**0x0132]**

DESCRIPTION

Nb-DISCONNECT indication advises the PE that the distant terminal has issued an Nb-DISCONNECT request. Upon receiving this indication, the PE discards any information it is holding for the current PEI-AEI pair and marks the PEI as available for reuse.

ADDITIONAL PARAMETERS

Optional:

Cause	- Reason for disconnecting
Distant Number	- Address being disconnected from
Distant Sub-address	- Sub-Address being disconnected from
User-User message	- A user specified message

5.4.1.1.9. **Nb-DISCONNECT** request [**0x0133]**

DESCRIPTION

Nb-DISCONNECT request is used to “hang up” an existing call or to reject an incoming call which is unwanted.

ADDITIONAL PARAMETERS

Optional:

Cause	- Reason for disconnecting
Local Number	- Address being disconnected from
Local Sub-address	- Sub-Address being disconnected from
User-User message	- A user specified message

5.4.1.1.10. **Nb-DISCONNECT** response [**0x0134]**

DESCRIPTION

Nb-DISCONNECT response is used to terminate a call in response to an incoming Nb-DISCONNECT indication. Upon issuing this response, the PE discards any information it is holding for the current PEI-AEI pair and marks the PEI as available for reuse.

ADDITIONAL PARAMETERS

Optional:

Cause	- Reason for disconnecting
Local Number	- Address being disconnected from
Local Sub-address	- Sub-Address being disconnected from
User-User message	- A user specified message

5.4.1.1.11. **Nb-ERROR** indication [**0x0141]**

DESCRIPTION

Nb-ERROR indication informs the PE that the AE has rejected a command which the PE passed across the interface.

ADDITIONAL PARAMETERS

Class — The class to which the error belongs. Possible error classes are:

- Resource - No resource(s) available
- Syntax - Syntactic error in the ASI message
- State - Illegal ASI message in current state

Code — A specific error value indicating the specific area of concern within the general class of errors.

Command — The command which caused (contained) the error.

Tag — The tag of invalid item associated with a **Syntax** error.

5.4.1.1.12. **Nb-EVENT** indication [**0x0151]**

DESCRIPTION

Nb-EVENT indication informs the PE that the AE has received some type of call progress indicator from the network. The PE should check the EVENT type and take any required action.

ADDITIONAL PARAMETERS

Event Type — The event type notifies the distant terminal of the type of activity occurring:

- Alerting — The user is being alerted (i.e., phone rings)
- Call Proceeding — Activity related to call setup is happening
- Notify — A change in bearer capability indicated
- Progress — Local terminal working on the call (play tones)
- Connect Ack — Indicates the successful completion of a connection

Cause — Reason for event

Display — Message from network

Progress Indicator — Progress of call

Signal — Audio/Visual alerting indicator

User-User — User specified message

5.4.1.1.13. **Nb-EVENT** request [**0x0152]**

DESCRIPTION

Nb-EVENT request is used to generate messages which inform distant terminal of ongoing activity at the local terminal. One example of this would be to let the distant terminal know that the local terminal is alerting the user.

ADDITIONAL PARAMETERS

Event Type	The event type notifies the distant terminal of the type of activity occurring:
- Alerting	The user is being alerted (i.e., phone rings)
- Call Proceeding	Activity related to call setup is happening
- Notify	A change in bearer capability indicated
- Progress	Local terminal working on the call (play tones)
- Connect Ack	Indicates the successful completion of a connection

Cause	Reason for event
Progress Indicator	Progress of call
User-User	User specified message

5.4.1.1.14. **Nb-MORE_INFO** indication [**0x0161]**

DESCRIPTION

Nb-MORE_INFO indication is a request from the network for the terminal to send or complete the called address information.

ADDITIONAL PARAMETERS

Required:	None

5.4.1.1.15. **Nb-MORE_INFO** response [**0x0162]**

DESCRIPTION

Nb-MORE_INFO response is used to send additional dial digits. The Nb-MORE_INFO response is a response to the Nb-MORE_INFO indication, and is used only when the terminal is in overlap sending mode.

ADDITIONAL PARAMETERS

Address	A block of data which contains the Address to be sent.

NOTE: The additional address information requested by the Nb-MORE_INFO indications may be sent in one or more Nb-MORE_INFO response(s).

5.4.1.1.16. **Nb-USER_USER_DATA** indication [**0x0171]**

DESCRIPTION

This indication is used to inform the PE of the arrival of a USER-USER message element via the USER INFORMATION message. USER-USER messages contained in other ISDN messages are delivered via their associated indications. Nb-USER_USER_DATA can only occur while the call is in the ACTIVE state.

ADDITIONAL PARAMETERS

Message	A block of data which contains the message to be placed in the USER-USER information element.

NOTE: This command is not supported by National ISDN-1 or ANS T1.607 and is currently included for Q.931 compatibility. This command is for further study.

5.4.1.1.17. **Nb-USER_USER_DATA** request [**0x0172]**

DESCRIPTION

The Nb-USER_USER_DATA request is used to send USER-USER messages to the distant terminal via the USER INFORMATION message. This request is only used when the connection is in the ACTIVE (connected) state. USER-USER messages can be sent during other phases of a call by including them as data elements in the appropriate requests and responses.

ADDITIONAL PARAMETERS

Message	A block of data which contains the message to be placed in the USER-USER information element.

NOTE: This command is not supported by National ISDN-1 or ANS T1.607 and is currently included for Q.931 compatibility. This command is for further study.

5.4.2. Supplementary Services

For future study. As contributions for this section become available, information will be included.

5.5. Management Plane Functions

5.5.1. Maintenance and Administration Services

DESCRIPTION

Maintenance and Administration services are available to a **PE** for performance of specific control over the local **AE**. Maintenance services govern operational sanity and functional testing, error reporting and general network management features of the **AE**. Administration services govern configuration, provisioning, service initialization and termination and capability reporting features of the **AE**.

5.5.1.1. Commands

This section contains the definitions and descriptions of the messages which are sent from the **PE** to the **AE**. R*equests* initiate actions in the AE such as generating reports or performing resets. C*onfirmations* are messages which conclude a previous request.

5.5.1.1.1. **Nb-AE_STATUS** confirmation [**0x0201]**

DESCRIPTION

Nb-AE_STATUS confirmation contains the current status of the **AE**. The status information that is returned is specific to the status of the **AE** operating system resources.

ADDITIONAL PARAMETERS

Report		A report of the AE resource status.

COMMENTS

The response to an **AE** Status command should include the following information.

- A list of all NAs (list of Adaptor_id primitive types).
- Identification of the AE.
- Vendor, version and revision.
- Active channels. A list of channels that are connected.
- A profile of the current AE resource utilization.
- Map of call appearances and associated sessions (session_id).
- Result of most recent AE sanity diagnostic.
- Information related to supplementary services.

Implementation of this command may include management information about all protocol stacks and events controlled by the **AE**.

5.5.1.1.2. **Nb-AE_STATUS** request [**0x0202]**

DESCRIPTION

Nb-AE_STATUS request requests the current status of the AE. The status information that is returned is specific to the status of the AE operating system resources. Upon receipt of AE Status, the AE will collect its current internal status information and format a response to the requesting PE.

ADDITIONAL PARAMETERS

Required:		None

5.5.1.1.3. **Nb-CAPABILITY** confirmation [**0x0211]**

DESCRIPTION

Nb-CAPABILITY confirmation is a confirmation from the AE listing the capabilities it was provisioned for.

ADDITIONAL PARAMETERS

Report		A block of information to be received from an AE.

COMMENTS

The response to an **Nb-CAPABILITY** confirmation command should include the following information and remains for further study.

- Types of channels supporting services (D, B1, B2, and/or H channels)
- Types of CSD supported and associated channel (clear/B1, v.120/B2, other.)

- Types of PSD supported and associated channel(provisioned/B1, POD_in/B2, POD_out/B2, --Dch...)
- Available RAM (in some basic unit that is processor independent)
- Simultaneous capabilities (CSV+CSD, CSD+CSD, CSD+CSD+DPKT...)
- Others are for further study

5.5.1.1.4. **Nb-CAPABILITY** request [**0x0212]**

DESCRIPTION

Nb-CAPABILITY request requests a report of the capabilities for which the AE has been provisioned.

ADDITIONAL PARAMETERS

Required: None

5.5.1.1.5. **Nb-CONFIGURE** request [**0x0221]**

DESCRIPTION

Nb-CONFIGURE request may be used by the PE to send configuration, and/or vendor specific, information to the AE. The information sent may be anything from simple strings, integers, and data structures, to executable code.

It is not within the scope of the ASI Issue 1 to specify an implementation for the configuration database. Possible implementations include Keys mapping to filenames, fields within a file, or data within the address space of the PE.

Future Issues of the ASI will address the system administration issues associated with managing the AE.

ADDITIONAL PARAMETERS

Adaptor Id: Unique name of the AE, this may be used by the PE to access a particular database if per interface databases are required.

Standard Key(s): This is tag number of a specific configuration data block to be followed by the data block.

- or -

Vendor Key(s): This is an ASCII encoded string to be followed by a vendor value data block.

Vendor Value: Value is a counted binary string with a maximum length bounded by the maximum size of an ASI message.

5.5.1.1.6. **Nb-CONFIGURE** confirmation [**0x0222]**

DESCRIPTION

Nb-CONFIGURE confirmation provides the status from a previous Nb-CONFIGURE request.

ADDITIONAL PARAMETERS

Status

A value which indicates the success or failure of the previous Nb-CONFIGURE request.

5.5.1.1.7. Nb-CONFIGURE indication [0x0223]

DESCRIPTION

Nb-CONFIGURE indication may be used by the AE to request configuration and/or vendor specific information from the PE. The information requested may be anything from simple strings, integers, and data structures, to executable code.

It is not within the scope of the ASI Issue 1 to specify an implementation for the configuration database. Possible implementations include Keys mapping to filenames, fields within a file, or data within the address space of the PE.

Future Issues of the ASI will address the system administration issues associated with managing the AE.

ADDITIONAL PARAMETERS

Adaptor Id

Unique name of the AE, this may be used by the PE to access a particular database, if databases are required per interface.

Standard Key(s)

This is tag number of a specific configuration data block.

Vendor Key(s)

This is an ASCII encoded string.

5.5.1.1.8. Nb-CONFIGURE response [0x0224]

DESCRIPTION

Nb-CONFIGURE response may be used by the PE to supply configuration and/or vendor specific information to the AE at the request of the AE.

It is not within the scope of the ASI Issue 1 to specify an implementation for the configuration database. Possible implementations include Keys mapping to filenames, fields within a file, or data within the address space of the PE.

Future Issues of the ASI will address the system administration issues associated with managing the AE.

ADDITIONAL PARAMETERS

AdaptorId

Unique name of the AE, this may be used by the PE to access a particular database, if databases are required per interface .

Standard Key(s)

This is tag number of a specific configuration data block to be followed by the data block.

- or -

Vendor Key(s)

This is an ASCII encoded string to be followed by a vendor value data block.

Vendor Value	Value is a counted binary string with a maximum length bounded by the maximum size of an ASI message.

5.5.1.1.9. **Nb-NA_INFO** confirmation [**0x0231]**

DESCRIPTION

Nb-NA_INFO confirmation contains the current status of the **AE** hardware. The status information that is returned is specific to the status of the **AE** hardware resources.

ADDITIONAL PARAMETERS

Adaptor Id	Id of adaptor selected via Nb-NA_INFO request
Report	A report of the AE resource status.

COMMENTS

The response to an **Nb-NA_INFO** command should include the following information and some items remains for further study.

Included:

- Adaptor ID
- NA Serial Number
- Vendor
- Firmware Version
- Hardware Version
- Self Test Results

For Further Study:

- Memory_utilization
- Errors
- Hook State
- Additional parameters

5.5.1.1.10. **Nb-NA_INFO** request [**0x0232]**

DESCRIPTION

Nb-NA_INFO request requests the current status of the NA(s) supported by the AE. The status information that is returned is specific to the status of the NA resources. Upon receipt of Nb-NA_INFO, the AE will collect status information from the appropriate NA(s) and format a response to the requesting PE.

ADDITIONAL PARAMETERS

Adaptor Id	A value which allows the PE to select the NA which it wants information about. Legal values for this field are obtained from the report returned by the Nb-AE_STATUS request. A value of **0xFFFF** will be used to indicate ALL.

COMMENTS

This command requires further study.

5.5.1.1.11. **Nb-RESET** confirmation [**0x0241**]

DESCRIPTION

Nb-RESET confirmation informs the AE about the success or failure of a previous Nb-RESET request.

ADDITIONAL PARAMETERS

Module — A component of the **AE** to which a specific reset was issued.

Result — A value indicating the success or failure of the reset.

5.5.1.1.12. **Nb-RESET** request [**0x0242**]

DESCRIPTION

Nb-RESET request provides mechanisms to accomplish different levels of reset to the AE. Reset can perform hard or soft resets of the entire AE system, or can reset only certain functions that may be operating on the adaptor without affecting other functional areas (i.e., a user data protocol can be reset without affecting signaling functions or calls in progress).

The ability to perform any reset mode operation depends upon the implementation and current state of the **AE** at the time the reset is issued. It is perfectly acceptable for the **AE** to not accept certain modes of the reset command. In multiuser system environment resets may be limited to privileged users.

ADDITIONAL PARAMETERS

Module — A component of the **AE** to which a specific reset may be issued (this information is derived from the AE capability report). A value of zero indicates the command pertains to the entire AE. Non-zero values indicate other functions within the **AE.**

r_mode — The reset mode to be invoked by the **AE**.

r_mode = 0 — Indicates the hardest level of reset. All **AE** hardware and software is completely reset and restarted (equivalent to a power up restart). This reset mode takes effect immediately, regardless of the other ongoing activities within the **AE**. All **AE** sessions are dropped and all ongoing communication is terminated.

r_mode = 1 — Indicates that all system software is reset and restarted. All existing calls are gracefully disconnected, all memory and communication buffers are cleared, and all **AE** software is restarted from its default initialized state. In this mode, the reset occurs when the last existing call terminates. No new calls are accepted while the reset is pending.

r_mode = 2 — Indicates an out of service condition for the function denoted by the module field. As currently active calls or sessions are concluded (in normal fashion) appropriate communication buffers will be cleared. Once all activity ceases the function will stop and not be restarted.

r_mode = 3 Indicates that the function denoted by the module field is to be reset and stopped immediately. All communication buffers of that function will be cleared. If the module is a protocol process any existing connections will be immediately torn down. The module is not restarted.

r_mode = 4 Indicates that the function denoted by the module field is to be reset and restarted. All communication buffers of that function will be cleared. If the function is a protocol process any existing connections will be gracefully torn down. The module is restarted to allow further transactions with the **PE**(s).

r_mode = 5 Indicates that the function denoted by the module is to be restarted to allow further transactions with the **PE**(s). Restart can be issued independently or subsequent to a r_mode = 3 reset operation. If issued independently, restart will clear all communication buffers and initialize the function to a known state.

5.5.2. Feature Access

For future study. As contributions for this section become available, information will be included.

5.6. ASI Errors

ASI errors are reported asynchronously via the Nb-ERROR indication. In order to help the PE manage the errors, errors are assigned two fields: CLASS and CODE.

The CLASS field is divided into three categories: Resource, Syntax, and State. Resource errors are defined as errors which stem from a request to use either: a) resources beyond the limits for which a particular AE subsystem has been provisioned, or b) a resource for which the AE has not been provisioned at all.

Syntax errors are defined as those errors which arise from the use of unspecified values anywhere in an ASI message. Upon detection of a syntax error, the AE will attempt to indicate the particular value which was in error by specifying the command and the tag which were in error.

State errors are those errors which arise from the issuing of a command in a sequence not accepted by the protocol in question. An example of this would be the issuing of an Nb-EVENT request with the event type set to ALERTING after the call has been connected. On state errors, the AE will return the command which caused the error.

The remainder of this section contains sample errors for each error class. A complete listing of errors is found in Chapter 6 - ASI Data Structures.

5.6.1. Resource Errors

The following is a sample list of resource errors. A complete listing of errors is found in Chapter 6 - ASI Data Structures.

- Unknown Resource Type
- Resource Not Accepting Requests
- No Resource(s) Available
- Resource Type Not Configured

5.6.2. Syntax Errors

The following is a sample list of syntax errors. A complete listing of errors is found in Chapter 6 - ASI Data Structures.

- Invalid Command
- Invalid Command Length
- Invalid Parameter Tag
- Invalid Parameter Length
- Invalid Parameter Value
- Invalid AEI
- Invalid PEI
- Invalid AEI-PEI pair
- Insufficient Parameters

5.6.3. State Errors

The following is a sample list of state errors. A complete listing of errors is found in Chapter 6 - ASI Data Structures.

- Not Valid During CONNECT phase
- Not Valid During DISCONNECT phase
- Not Valid During ACTIVE phase
- Not Valid During IDLE phase

6.0. ASI Data Structures

6.1. Introduction

This chapter defines the content of ASI data blocks. Data blocks are data structures associated with those ASI commands that transfer information in one or both directions across the ASI interface. In order to simplify the presentation of the ASI data blocks, primitives and constructed data structures are defined and used as building blocks. This chapter deals only with those commands that have associated data blocks; a complete description of ASI commands is provided in Chapter 5.

All data blocks are passed across the ASI interface in the Additional Parameters (AP) field of an ASI command's message. Each ASI command is assigned a Command Name (two octets) and Length (two octets) in the fixed message header. The Length indicates the size of the Additional Parameters field. Additional Parameters are constructed out of one or more data blocks and each data block containing one or more parameters. If a command does not have an associated data block, then its Length is zero.

The ASI data blocks are classified into three distinct types which are: Session Blocks, the Configuration Block and General Data Blocks. Session Blocks are call dependent, the Configuration Block is environment dependent and the General Data Blocks are command dependent.

In addition to parameters normally associated with a command, the ASI also defines Independent data blocks which are global, transient, always optional and can be added to the contents of a number of different commands in the AP field. Presently the only identified Independent data block is the user_to_user data block. A command's length parameter indicates the length of the total contents which includes its associated parameters and any Independent data block.

An ASI data block contains a list of parameters, some of which can be primitive, constructed and/or independent types. These are defined and encoded using ASN.1. Other parameters are also used within a data block; they are defined and encoded using ASN.1 concepts but do not fully conform to the standard. This ASI specific encoding method was adopted to provide flexible, yet efficient interpretation of the various data structures. The encoding of the data structures presented throughout this chapter will be easily understood by those familiar with ASN.1. In addition, Section 6.6 provides some typical examples of encoded ASI data structures. The examples are provided in tabular form, showing tag (name), length, and value for each parameter in the data structures.

A number of the ASI parameters are not required in all circumstances and these have been labeled with “*” at the end of their definition. The “*” is not a formal part of the encoding notation. The ASN.1 OPTIONAL notation is not used in this document. While there are optional parameters in ASI data blocks that may be supplied in an AE implementation, the value supplied by the PE takes priority.

The rest of this chapter is organized as follows.

Section 6.2 defines common data structures: *primitive*, *constructed* and *independent* data blocks.

Section 6.3 covers Session Blocks.

Section 6.4 covers the Configuration Block.

Section 6.5 covers the General Data Blocks.

Section 6.6 covers examples of data block encodings.

Issue I of the ASI specification provides the capabilities required for Basic Voice, D Channel Packet Data and B Channel (nailed-up) Packet Data Calls. While considerable work has been completed on the data structures for other types of calls and Management Plane functions these should not be considered to be completed. The notations “for further study” or “for future study” are used to indicate those sections and areas that will be completed in future Issues of this document.

6.2. Common Data Structures

The common data structures are building blocks for other data structures. They are used as component parts of the ASI data blocks defined in later sections of this chapter. The three types of common data structures are defined in this section: primitive, constructed and independent data blocks.

6.2.1. Primitive Data Types

These data types are used as building blocks for constructed data types and ASI data block definitions.

```
Adaptor_id ::= [PRIVATE 47] IMPLICIT INTEGER SIZE (2)

   -- hex value X'df2f'

Addr_digits ::= [PRIVATE 31] IMPLICIT PrintableString

   -- hex value X'df1f'


Addr_type ::= [PRIVATE 32] IMPLICIT OCTET {

   -- hex value X'df20'
   -- see T1.607  4.5.13
   -- Code as bits 5-7 of octet 3,
   -- (Current T1.607 values: 00,10,20,30,40 & 60 hex)
   -- also see the Numbering_plan primitive.}

Block_type ::= [PRIVATE 33] IMPLICIT ENUMERATED {

   --  hex value X'df21'
   --  Defines the type of Session Block
   --  See Section 6.3.1.1 for definition of call owner and user
   voice(1),                -- voice call
   psd(2),                  -- provisioned PSD (B or D channel)
   csd_clear_chan(3),       -- no rate adaption
   csd_v120_owner(4),       -- V.120, call owner
   csd_v120_user(5),        -- V.120, call user
   csd_x25_owner(6),        -- X.25 end-to-end or to a PSN, call owner
   csd_x25_user(7),         -- X.25 end-to-end or to a PSN, call user
   csd_i515(8),             -- I.515 negotiated rate adaption
                            -- csd_i515 is for further study
   csd_v110(9),             -- V.110, for further study
   pod_owner(10),           -- packet on demand
   pod_user(11)             -- packet on demand }

   -- additional session block types are for further study.
```

```
Call_type ::= [PRIVATE 34] IMPLICIT ENUMERATED {

    -- hex value X'df22'
    -- used in PSD session blocks

    svc(0),                  -- switched virtual circuit two way access
    svc_in(1)                -- switched virtual circuit incoming access only
    svc_out(2),              -- switched virtual circuit outgoing access only
    pvc(3)                   -- permanent virtual circuit }


Cause ::= [PRIVATE 35] IMPLICIT INTEGER {

    -- hex value X'df23'
    -- See T1.607, 4.5.11. for full list of cause codes which include both
    -- class and value .  Use the values from the Cause Table
    -- (bits 1-7) for the decimal numbers below.
    -- see also the Diag primitive type.

    unassigned number(1),
    -- ...
    interworking, unspecified(127) }


Channel_type ::= [PRIVATE 36] IMPLICIT INTEGER (0..30)

    -- HEX VALUE x'df24'
    -- The value of channel_type is:
    --      0: for D channel
    --      1: for B channel
    --      6: for H0 channel
    --     23: for H10 channel
    --     24: for H11 channel
    --      n: for any other nx64 channels



Cug_type ::= [PRIVATE 37] IMPLICIT INTEGER {

    -- hex value X'df25'
    -- see X.25, 6.14

    no_cug(0),
    cug_ia(1),               -- CUG with Incoming Access
    cug_oa(2),               -- CUG with Outgoing Access
    cug_io(3),               -- CUG with Both Incoming &  Outgoing  Access
    cug_pref(4),             -- preferential CUG
    cug_ia_pref(5),          -- preferential CUG with Incoming Access
    cug_oa_pref(6),          -- preferential CUG with Outgoing Access
    cug_io_pref(7)           -- preferential CUG with Bilateral Access }
```

```
Diag ::= [PRIVATE 48] IMPLICIT INTEGER {

    -- hex value X'df30'
    -- See T1.607, 4.5.11. for full list of diagnostic codes
    -- Use the decimal values of bits 1-7.
    -- also see the Cause primitive type.

    -- this primitive is for further study.}


Display ::= [PRIVATE 38] IMPLICIT PrintableString

    -- hex value X'df26'
    -- see T1.607, 4.5.15. Octet 3.



Num_stops ::= [PRIVATE 51] IMPLICIT ENUMERATED  {
 -- hex value X'df33'

one_plus_stop (0),           -- one and a half stop bits
one_stop (1),                -- one stop bit
two_stop (2)                 -- two stop bits }

Num_data_bits ::= [PRIVATE 52] IMPLICIT ENUMERATED {

    -- hex value X'df34'
    -- see V.120 spec
    -- number of bits including parity if present.

    five_bits (0),
    six_bits (1),            -- this value is not defined in V.120 (for future use)
    seven_bits (2),
    eight_bits (3) }

Numbering_plan ::= [PRIVATE 39] IMPLICIT OCTET {

    -- hex value X'df27'
    -- see T1.607, 4.5.13.
    -- Code as value of octet 3, bits 1-4
    -- current T1.607 values (00,01,03,04 & 05 hex)
    -- see also Addr_type primitive.}


Parity_type ::= [PRIVATE 53] IMPLICIT ENUMERATED {

    -- hex value X'df35'
    -- see V.120 spec

    odd (0),
    even (1),
    none (2),
    force_zero (3),
    force_one (4) }
```

```
Presentation_ind ::= [PRIVATE 40] IMPLICIT OCTET {

    -- hex value X'df28'
    -- see T1.607, 4.5.13.
    -- code as shown for octet 3a bits 6-7
    -- the following are the present T1.607 parameters in hex.

    allowed(00),
    restricted(20),
    number not available(30) }


Presentation_num_digits ::= [PRIVATE 49] IMPLICT PrintableString {

    -- hex value X'df31'
    -- see T1.607, 4.5.13 }

Rate ::= [PRIVATE 41] IMPLICIT OCTET {

 -- hex value X'df29'
 -- see T1.607 4.5.5 octet 5a for coding, These code points are identical
 -- those specified in the V.120 Low Layer Compatibility IE. }

Saddr_type ::= [PRIVATE 46] IMPLICIT OCTET {

    -- hex value X'df2e'
    -- see T1.607  4.5.10 & 4.5.14
    -- Code as bits 5-7 of octet 3,
    -- (Current T1.607 values: 00 & 10 hex) }

Saddr_info :: [PRIVATE 50] IMPLICIT SET {

    -- hex value X'df32'
    -- see T1.607, 4.5.10 octet 4
    -- this primitive is for further study }

Screening_ind ::= [PRIVATE 42] IMPLICIT OCTET {

    -- hex value X'df2a'
    -- see T1.607, 4.5.13.
    -- code as shown for octet 3a bits 1-2
    -- the following are the present T1.607 parameters in hex.

    udef(00),                -- user-provided, not screened
    udef_verif(01),          -- user-provided, verified and passed
    udef_invalid(02),        -- user-provided, verified and failed
    nprov(03)                -- network provided }


Session_id ::= [PRIVATE 43] IMPLICIT OCTET STRING SIZE (4)

    -- hex value X'df2b'
    -- aei: the first two octets.
    -- pei: the next two octets
    -- Octet order is High followed by Low
```

```
Signal ::= [PRIVATE 44] IMPLICIT INTEGER {

   --  hex value X'df2c'
   --  see T1.607, 4.5.24. octet 3
   --  use decimal equivalent of binary codes in T1.607.

   dial_tone_on(0),
   ring_back_tone_on(1),
   -- ...
   alerting_off(79) }


Xfer_cap ::= [PRIVATE 45] IMPLICIT INTEGER {

   -- hex value X'df2d'
   -- see T1.607, 4.5.5.
   -- code as shown for octet 3, bits 1-5
   -- the following are the present T1.607 parameters.

   speech(0),
   unrestricted_digital(8),
   restricted_digital(9),
   3.1 kHz audio(16),
   7 kHz audio(17) }        -- known as “multi-use bearer” in ANSI
```

6.2.1.1. Context Specific Primitive Data Types

The following primitive data types are context specific, thus there are no tag numbers assigned.

```
Call_state ::= ENUMERATED {

   unsupported(0),
   alerting (1),
   connected (2),
   connected association (3),
   connect request (4),
   dialing (5),
   held (6),
   held association (7),
   held conferencing (8),
   held transfer (9),
   idle (10),
   -- additional parameters are for further study
   }

Codec_law ::= ENUMERATED {

   mu_law (0),
   a_law (1) }


Device ::= ENUMERATED {

   analog_cs (0),           -- analog circuit switched
   digital_cs (1) }         -- digital circuit switched
```

```
Event_type ::= INTEGER {

   alerting (0),
   call_proceeding (1),
   notify (2),
   progress (3),
   connect_ack(4) }


Error_class ::=  ENUMERATED {

   resource (0),
   syntax (1),
   state(2) }


Location ::= ENUMERATED {

   user (0),
   local_private_net (1),
   local_pub_net (2),
   transit_net (3),
   remote_pub_net (4),
   remote_priv_net (5),
   outside (6) }


Protocol_opt ::= ENUMERATED {

   -- used in V.120 to set mode of operation to asynch, synch or transparent.

   asynch(0),
   synch (1),
   transp (2) }


Seq_num_mod ::= ENUMERATED {

   modulo_8 (8),
   modulo_128 (128) }


X25_clear_cause ::= INTEGER {

   -- use the codes from Table 20/x.25 in Section 5.2.4.1.1
   -- of the CCITT 1988 Rec. X.25
   -- In the following r_* indicates a cause indicated by the remote DTE
   -- and l_* by the local DTE.
   l_number_busy (1),
   .
   .
   l_rpoa_out_of_order (21),
   r_number_busy (129),
   .
   .
   r_rpoa_out_of_order (149) }
```

```
X25_diag ::= INTEGER {

   -- use the diagnostic codes specified by the CCITT 1988 X.25
   -- recommendation }

X25_reset_cause ::= INTEGER {

   -- In the following r_* denotes a cause indicated by the remote DTE
   -- and l_* by the local DTE.  The coding is from the CCITT 1988
   -- Rec. X.25 Section 5.4.3.1.  The reset cause is carried in the
   -- X.25 reset indication packet.

   l_out_of_order (1),       -- pvc only
   l_remote_proc_err (3),
   l_local_proc_err (5),
   l_net_congestion (7),
   l_remote_dte_op (9),      -- pvc only
   l_net_op (15),            -- pvc only
   l_incompatible_dest (17),
   l_net_out_of_order (29)   -- pvc only
   r_out_of_order (129),     -- pvc only
   r_remote_proc_err (131),
   r_local_proc_err (133),
   r_net_congestion (135),
   r_remote_dte_op (137),    -- pvc only
   r_net_op (143)            -- pvc only
   r_incompatible_dest (145),
   r_net_out_of_order (157)  -- pvc only }


X25_restart_cause ::= INTEGER {

   -- The following coding is from the CCITT 1988
   -- Rec. X.25 Section 5.5.1.1.  The restarting cause is carried in the
   -- X.25 restart indication packet.

   network_congestion (2),
   network_operational (7),
   registration_cancellation_confirmed (127) }
```

6.2.2. Constructed Data Types

The constructed data types are, in part, built from the primitive data types defined in Section 6.2.1. Like the primitive data types, these structures are also used to define the data structures of the various types of ASI data blocks.

```
Channel ::= [PRIVATE 78] IMPLICIT SET {

   -- hex value X'ff4e'

   channel_type   Channel_type,

   -- The range of channel_num is as follows:
   --   BRI: (0..2)
   --   PRI: (0..23) or (0..30)
   -- If the channel_type = 0 (D channel), channel_num should be 0.
   -- If the channel_type is not D channel, a value of 0 in the
   -- channel_num field indicates “any channel”.
   -- The encoding for Non-Facility Associated Signaling (NFAS)
   -- B channels is for further study.
   channel_num    [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER; }

Directory_number ::= SET {

   -- context specific structure

   addr_type                Addr_type,
   numbering_plan           Numbering_plan,
   addr_digits              Addr_digits,
   s_ind                    Screening_ind, *
   p_ind                    Presentation_ind, *
   p_num                    Presentation_num_digits * }


Progress_ind ::= SET {

   location                 [CONTEXT-SPECIFIC 31] IMPLICIT Location,
   description              [CONTEXT-SPECIFIC 32] IMPLICIT INTEGER (1..10) }


Subaddress ::= SET {

   -- context specific structure
   -- for further study

   addr_type                Saddr_type,
   addr_info                Saddr_info }
```

6.2.3. Independent Data Blocks

Independent data blocks are global, transient, always optional and can be added to the contents of more than one command. At the present time only one Independent data block, user_to_user, has been identified.

```
User_to_user ::= [PRIVATE 127] IMPLICIT OCTET STRING

   -- hex value X'df7f'
```

6.3. Session Blocks

6.3.1. General

Each active call (session) has an associated Session Block which is used to define all of the call related information required by both the AE and the PE. The Session Block is associated with the Nb-CONNECT request, Nb-CONNECT confirmation, Nb-CONNECT indication and Nb-CONNECT response commands.

For an outgoing call an Nb-CONNECT request is issued with the Session Block specified in the Additional Parameter field. The AE automatically opens a session and upon successful completion of the call sends the PE an Nb-CONNECT confirmation which can include modified or additional session block parameters. If the call is not successfully completed, then the PEI is sufficient to identify the failed session to the PE via an Nb-DISCONNECT indication or an Nb-ERROR indication.

For an incoming call the AE creates a session block and populates parameters from the information provided by the network in the setup message. The session block is then sent to the PE by an Nb-CONNECT indication. The PE responds with an Nb-CONNECT response which can provide the AE with any missing session block information. The Nb-CONNECT response is required but if the PE does not need to send any additional parameters, then the session block can be empty. The ASI does not allow default parameter values to be supplied by the AE.

Currently 11 types of ASI session blocks have been identified. These are defined by the use of a unique Block Type Parameter for each session. The parameters associated with each type of Session Block are provided in the remainder of this section. A number of these Session Block types are for further study.

Session block types

----

* voice call

* provisioned Packet Switched Data (PSD) (B or D channel)

* Circuit Switched Data (CSD), Clear channel (no rate adaption)

* CSD, V.120, call owner

* CSD, V.120, call user

CSD, X.25 end-to-end or to a PSN, call owner

CSD, X.25 end-to-end or to a PSN, call user

I.515 negotiated rate adaption

V.110

Packet on Demand (POD), call owner

POD, call user

Note: The above session blocks that are marked with “*” are currently defined. The remaining session blocks are for further study.

6.3.1.1. Definition of Call owner and Call User

The following describes ASI "call owner" and "call user" sessions which are used where required throughout this chapter.

Protocols that are implemented in the AE that allow multiple virtual circuits over a Circuit Switch Data (CSD) connection may require the creation of multiple ASI Sessions. For example, X.25 over a CSD connection and V.120 CSD are ASI sessions where this requirement has been identified. The session that sets up the CSD call (originates or answers a call) with X.25 (CSD) or V.120 rate adaption is known as the "call owner". Call owner sessions create the end-to-end transport but do not carry data from PE to PE.

For each virtual circuit required for PE to PE data transport, a separate session is created. The ASI's nomenclature for this type of session is a "call user" session. While there can be only one call owner session associated with a B-

Channel, each call owner session can have many associated call user sessions. The AE must reject an Nb-DISCONNECT request to an owner session if there are one or more associated user sessions active. The cause returned by Nb-ERROR indication should be "not_valid_ active_phase (33)".

Note: The PE may not bind a user plane for any call owner session.

6.3.1.2. Definition of Local and Distant

Local and distant are used instead of calling and called to describe directory numbers and sub-addresses. The local_dir_num is the calling DN for outgoing calls and the called DN for incoming calls. The distant_dir_num is the called DN for outgoing calls and the calling DN for incoming calls.

6.3.2. Circuit Switched Voice (CSV) Session Block

This session block provides the parameters required for basic Circuit Switched Voice calls. Additional parameters required for support of ISDN Supplementary Services are for future study.

```
block_type              Block_type, -- type = 1
local_dir_num           [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number, *
local_sub_addr          [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *
distant_dir_num         [CONTEXT-SPECIFIC 33] IMPLICIT Directory_number, *(1)
distant_sub_addr        [CONTEXT-SPECIFIC 34] IMPLICIT Subaddress, *
xfer_cap                Xfer_cap,
channel_prefer          Channel,
call_appearance         [CONTEXT-SPECIFIC 35] IMPLICIT INTEGER, *
```

Note: This is optional because there are some cases of an outgoing call when the DN may be provided by an adjunct device or the mode of operation is overlap sending.

6.3.3. Provisioned Packet Switched Data Session Block

This session block is used for provisioned packet switched data service, on a D or B channel. Suggested values are provided for some elements of this session block as guidance to applications programmers. These values are not intended to be interpreted as either mandatory or default values.

```
block_type               Block_type, -- type = 2
local_dir_num            [CONTEXT-SPECIFIC 31]  IMPLICIT Directory_number,
local_sub_addr           [CONTEXT-SPECIFIC 32]  IMPLICIT Subaddress, *
distant_dir_num          [CONTEXT-SPECIFIC 33]  IMPLICIT Directory_number,
distant_sub_addr         [CONTEXT-SPECIFIC 34]  IMPLICIT Subaddress, *
line_channel             Channel,

-- suggested value{65535}
transit_delay            [CONTEXT-SPECIFIC 35] IMPLICIT INTEGER (0..65535),
call_deflection_addr     Addr_digits, *
rpoa_name                [CONTEXT-SPECIFIC 36] IMPLICIT PrintableString, *
network_uid              [CONTEXT-SPECIFIC 37] IMPLICIT PrintableString, *

-- suggested value{0}, see CCITT X.25 1988, Table 30/X.25
xmit_thruput_class       [CONTEXT-SPECIFIC 38] IMPLICIT INTEGER (0..15),

-- suggested value{0}
rcv_thruput_class        [CONTEXT-SPECIFIC 39] IMPLICIT INTEGER (0..15),

-- suggested value{FALSE}
rev_charge_accepted      [CONTEXT-SPECIFIC 40] IMPLICIT BOOLEAN,

-- suggested value{FALSE}
rev_charge_requested     [CONTEXT-SPECIFIC 41] IMPLICIT BOOLEAN,
fast_select              [CONTEXT-SPECIFIC 42] IMPLICIT BOOLEAN, *
cug_name                 [CONTEXT-SPECIFIC 43] IMPLICIT PrintableString,  *
cug_type                 Cug_type, *

-- suggested value{FALSE}
d_bit_enabled            [CONTEXT-SPECIFIC 44] IMPLICIT BOOLEAN,
call_type                Call_type,

-- the value of pcv_num does not matter if call type is 0
-- suggested value{0}
pvc_num                  [CONTEXT-SPECIFIC 45] IMPLICIT INTEGER (0..4095), *

-- suggested value{8}
xmit_packet_size         [CONTEXT-SPECIFIC 46]  IMPLICIT  INTEGER (4..12),

-- suggested value{8}
rcv_packet_size          [CONTEXT-SPECIFIC 47]  IMPLICIT  INTEGER (4..12),

-- suggested value{2}
xmit_window_size         [CONTEXT-SPECIFIC 48]  IMPLICIT  INTEGER (1..7),

-- suggested value{2}
rcv_window_size          [CONTEXT-SPECIFIC 49]  IMPLICIT  INTEGER (1..7),

Q_bit_enabled            [CONTEXT-SPECIFIC 50] IMPLICIT BOOLEAN,

m_bit_enabled            [CONTEXT-SPECIFIC 51] IMPLICIT BOOLEAN,
   exp_data_enabled          [CONTEXT-SPECIFIC 52] IMPLICIT BOOLEAN,
```

6.3.4. Circuit Switched Data (CSD) Session Blocks

6.3.4.1. Circuit Switched Data Session Block for Clear Channel

```
   block_type               Block_type, -- type = 3
   local_dir_num            [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number, *
   local_sub_addr           [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *
   distant_dir_num          [CONTEXT-SPECIFIC 33] IMPLICIT Directory_number, *(1)
   distant_sub_addr         [CONTEXT-SPECIFIC 34] IMPLICIT Subaddress, *
   xfer_cap                 Xfer_cap,
   channel_prefer           Channel,
```

Note: This is optional because there are some cases of an outgoing call when the DN may be provided by an adjunct device or the mode of operation is overlap sending.

6.3.4.2. V.120 Rate Adaption -- Control Information Data Structure.

V.120 provides provisions for passing control information end-to-end and the ASI provides a way to pass this information between the AE and PE in the User Plane. The following defines the control buffer format for V.120 connections. This control buffer is passed on the User Plane synchronously with the data.

The format is as follows:

OCTET	DEFINITION
0	Local header length (H+CS+CS Extension)
1	Local Header (H)
2	Local Control State (CS)
3 to n	CS Extension, if any, as defined in ANSI V.120
n+1	Remote header length (H+CS+CS Extension)
n+2	Remote Header (H)
n+3	Remote Control State (CS)
n+4 to end	CS Extension, if any as defined in ANSI V.120

Note 1: If an ASI Entity (AE) makes unsolicited changes to any of V.120 header information it must convey the updated control buffer to the other entity immediately. The updated Control buffer is to be sent to the other entity, using an Access Method specific technique, even when there is no data to be sent.

Note 2: An entity may ONLY change the value of the local header. The remote header is for reference only and cannot be changed locally.

Note 3: Ability to change flag bits in the local header is dependent upon the bit mask received in a V.120 capabilities call. Only bits specified in the capabilities list will be passed down the interface.

Note 4: Remote header length field is mandatory, but may be set to 0.

Note 5: Header lengths do not include the header length octet.

6.3.4.3. Circuit Switched Data (CSD) Session Blocks, V.120 Rate Adaption

There are two types of V.120 Sessions an Owner Session and a User Session. See Section 6.3.1.1 for the definition of Call Owner and Call User.

6.3.4.3.1. CSD Session Block, V.120 Rate Adaption, Call Owner

```
-- The Call Owner session does not carry information for logical link
-- connections other than LLI 0.  If a specific implementation of an AE
-- also creates a user data LLI, the AE will keep this LLI active until
-- the PE requests a user session or the Call Owner session is
-- disconnected.

block_type                Block_type,  -- type 4
local_dir_num             [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number,*
local_sub_addr            [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *
distant_dir_num           [CONTEXT-SPECIFIC 33] IMPLICIT Directory_number,* (1)
distant_sub_addr          [CONTEXT-SPECIFIC 34] IMPLICIT Subaddress, *
xfer_cap                  Xfer_cap,
channel_prefer            Channel
```

Note: This is optional because there are some cases of an outgoing call when the DN may be provided by an adjunct device or the mode of operation is overlap sending.

6.3.4.3.2. CSD Session Block, V.120 Rate Adaption, Call User

```
-- See Section 6.3.1.1 for definition of call user.
-- Call User sessions are the only V.120 sessions that can pass data
-- and control information in the User Plane.

block_type                Block_type,  -- type 5
call_owner                Session_id,  -- the session ID of call owner

baud_rate                 Rate,        -- V.120 session baud rate

-- Include optional header for control info (CS) (Y/N)
cs_header                 [CONTEXT-SPECIFIC 35] IMPLICIT BOOLEAN,

-- if flag_fill is false then the interframe fill is all 1's
flag_fill                 [CONTEXT-SPECIFIC 36] IMPLICIT BOOLEAN,

-- The following specify the maximum number of octets allowed in a
-- frame. The size of 2 integers allows for values up to 65535 to be
-- specified.  The need to specify an upper limit is for further study.
max_xmt_frame_s           [CONTEXT-SPECIFIC 37] IMPLICIT INTEGER SIZE (2),
max_rcv_frame_s           [CONTEXT-SPECIFIC 38] IMPLICIT INTEGER SIZE (2),

-- The following selects the layer 2 operational mode.
-- True = multiple frame mode, False = unacknowledged-only mode (UI
-- frames only).
mult_frame                [CONTEXT-SPECIFIC 39] IMPLICIT BOOLEAN,

-- The Following Sets V.120 Operational Mode to Asynchronous,
-- Synchronous HDLC with octet aligned messages or Bit Transparent.
-- Support of Synchronous non-octet aligned messages is for further
-- study in ANSI V.120.
mode                      [CONTEXT-SPECIFIC 40] IMPLICIT Protocol_opt, -- See 6.2.1.1

-- ASYNCHRONOUS MODE PARAMETERS
num_data_bits             Num_data_bits,

parity_type               Parity_type,

-- The number of stop bits is sent end-to-end during initial hand shake.
-- It is not part of control information sent in the header.
num_stops                 Num_stops, *

-- OPTIONAL CONTROL PARAMETERS (ASYNCHRONOUS MODE ONLY)
-- For the following four parameters if the AE does not support the
-- parameter and the PE has specified a value in a (PE -> AE) Session
-- Block, then an Nb-ERROR indication with a
-- "Not Supported" cause will be returned by the AE.

-- If the AE does support one of these four parameters but the
-- PE does not specify a value, then the AE will use the default
-- value (and report the default value back to the PE in the next
-- message that carries a Session Block).
-- The only legal ASI default values are those designated by the ASI
-- Configuration facilities or those designed in by the CPE vendor.
-- (Also see 6.1)
```

```
    -- The following determine frame forwarding criteria for frames
    -- smaller than the maximum size (max_xmit_frame_s).

    -- Characters can be 5,7, or 8 bits with bit padding per ANSI V.120. If
    -- more than one character is desired then multiple instances of this
    -- parameter (data_forward_char) should be used.
    data_forward_char          [CONTEXT-SPECIFIC 41] IMPLICIT OCTET,*

    -- For the following timers, the value of 0 will indicate that no data
    -- forwarding on time-out is required; a value between 1 and 255 will
    -- indicate the value of the delay in twentieths of a second. The data
    -- accumulation timer forwards on the time-out initiated by the first
    -- character.  The idle timer is reset by each transmitted character
    -- and forwards upon time-out.
    data_accum_timer           [CONTEXT-SPECIFIC 42] IMPLICIT OCTET (0..255),*
    idle_timer                 [CONTEXT-SPECIFIC 43] IMPLICIT OCTET (0..255),*

    local_echo                 [CONTEXT-SPECIFIC 44] IMPLICIT BOOLEAN, *



    -- SYNCHRONOUS HDLC MODE PARAMETERS

    -- None identified that are specific to this mode.


    -- TRANSPARENT MODE PARAMETERS

    -- Discard data in frames with invalid CRC (T/F)
    bad_frame_reject           [CONTEXT-SPECIFIC 45] IMPLICIT BOOLEAN,

    -- Use Header (Y/N) (H is optional in this mode)
    header                     [CONTEXT-SPECIFIC 46] IMPLICIT BOOLEAN

    -- The need for a parameter to select D_Channel or in-band parameter
    -- negotiation is for further study.
    -- The parameters included in the V.120 Low Layer Compatibility
    -- Information Element's octets 6 and 7 are for further study.
    -- other parameters are for further study.}
```

6.3.4.4. Circuit Switched Data Session Block with X.25 End-to-End or to a Packet Switched Network other than ISDN, by Call Owner

```
    -- The following is for further study.
    block_type                Block_type, -- type = 6
    local_dir_num             [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number, *
    local_sub_addr            [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *
    distant_dir_num           [CONTEXT-SPECIFIC 33] IMPLICIT Directory_number, *(1)
    distant_sub_addr          [CONTEXT-SPECIFIC 34] IMPLICIT Subaddress, *
    xfer_cap                  Xfer_cap,
    channel_prefer            Channel,
```

Note: This is optional because there are some cases of an outgoing call when the DN may be provided by an adjunct device or the mode of operation is overlap sending.

6.3.4.5. Circuit Switched Data Session Block with X.25 End-to-End or to a Packet Switched Network, by Call User

```
-- The following is for further study.
block_type                Block_type, -- type 7
call_owner                Session_id,
```

6.3.4.6. Circuit Switched Data Session Block with I.515 Negotiation

```
-- The following is for further study.
block_type                Block_type,  -- Type 8
local_dir_num             [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number, *
local_sub_addr            [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *
distant_dir_num           [CONTEXT-SPECIFIC 33] IMPLICIT Directory_number, *(1)
distant_sub_addr          [CONTEXT-SPECIFIC 34] IMPLICIT Subaddress, *
xfer_cap                  Xfer_cap,
channel_prefer            Channel,
```

Note: This is optional because there are some cases of an outgoing call when the DN may be provided by an adjunct device or the mode of operation is overlap sending.

6.3.5. Packet On Demand (POD) Session Blocks

A POD owner session is one in which a circuit switched connection to the ISDN switch's packet handler is temporarily established. Calls on the virtual circuits (users) are then established using separate sessions. The session block for a user session is defined below. The session block for the owner session is for further study. Suggested values are provided for some elements of this session block as guidance to applications programmers. These values are not intended to be interpreted as either mandatory or default values.

6.3.5.1. Packet On Demand Session Block, Call Owner

```
-- For further study


block_type                Block_type, -- type = 10
-- additional parameters for further study
```

6.3.5.2. Packet On Demand (POD) Session Block, Call User

A POD user session is one that uses a POD owner facility. The session block includes the identification, in the parameter call_owner, of the owner session. If the owner session is disconnected then all user sessions will be lost.

```
block_type                Block_type, -- type 11
local_dir_num             [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number,
local_sub_addr            [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *
distant_dir_num           [CONTEXT-SPECIFIC 33] IMPLICIT Directory_number,
distant_sub_addr          [CONTEXT-SPECIFIC 34] IMPLICIT Subaddress, *
call_owner                Session_id,

-- suggested value{65535}
transit_delay             [CONTEXT-SPECIFIC 35] IMPLICIT INTEGER (0..65535),
call_deflection_addr      Addr_digits, *
rpoa_name                 [CONTEXT-SPECIFIC 36] IMPLICIT PrintableString, *
network_uid               [CONTEXT-SPECIFIC 37] IMPLICIT PrintableString, *

-- suggested value{0}
xmit_thruput_class        [CONTEXT-SPECIFIC 38] IMPLICIT INTEGER(0..15),

-- suggested value{0}
rcv_thruput_class         [CONTEXT-SPECIFIC 39] IMPLICIT INTEGER(0..15),

-- suggested value{FALSE}
rev_charge_accepted       [CONTEXT-SPECIFIC 40] IMPLICIT BOOLEAN,

-- suggested value{FALSE}
rev_charge_requested      [CONTEXT-SPECIFIC 41] IMPLICIT BOOLEAN,

fast_select               [CONTEXT-SPECIFIC 42] IMPLICIT BOOLEAN, *
cug_name                  [CONTEXT-SPECIFIC 43] IMPLICIT PrintableString, *
cug_type                  Cug_type,

-- suggested value{FALSE}
d_bit_confirmation        [CONTEXT-SPECIFIC 44] IMPLICIT BOOLEAN,
call_type                 Call_type,

-- suggested value{0}
-- the value of pcv_num does not matter if the call_type is 0.
pvc_num                   [CONTEXT-SPECIFIC 45] IMPLICIT INTEGER(0..4095), *

-- suggested value{8}
xmit_packet_size          [CONTEXT-SPECIFIC 46] IMPLICIT INTEGER(4..12),

-- suggested value{8}
rcv_packet_size           [CONTEXT-SPECIFIC 47] IMPLICIT INTEGER(4..12),

-- suggested value{2}
xmit_window_size          [CONTEXT-SPECIFIC 48] IMPLICIT INTEGER(1..7),

-- suggested value{2}
rcv_window_size           [CONTEXT-SPECIFIC 49] IMPLICIT INTEGER(1..7)
```

6.4. Configuration Block

The Configuration Block is associated with the Nb-CONFIGURE request, indication and response commands. The data block for Nb-CONFIGURE confirmation is described in Section 6.5.1.3. The commands that include the configuration block are members of a class of Administration and Maintenance (A&M) Commands that are transported over the ASI interface in the Management Plane. Commands that include the configuration block differ from the other members of the A&M class of commands in that they define the operational environment and a configuration block must be sent to the AE before any calls can be established. Configuration block parameters are global and static in nature as opposed to call or session related.

The contents of this data block are for further study and implementations built to the current ASI document are allowed to use vendor specific configuration techniques in addition to or instead of the ASI CONFIGURE commands. The presently identified configuration parameters are presented in this section. Standard and vendor keys are for further study.

A number of Configuration Block parameters are associated with the types of service the end user ordered from the ISDN service provider. These are called provisioned parameters in this document and may vary with the service providing switch and that switch's generic. Other parameters will contain vendor specific configuration information.

The Configuration Block encoding follows. A Configuration Block consists of one or more members of the following set of individual data structures which are defined in the remainder of this section. Additional members of this set are for further study.

```
    dx25_cb                  DX25_CB,*
    bx25_cb                  BX25_CB,*
    voice_cb                 VOICE_CB,*
    d_lapd_cb                D_LAPD_CB,*
    basic_cb                 BASIC_CB,*
    pri_cb                   PRI_CB,*                -- for further study
    vendor                   VENDOR_CB,*             -- for further study.
```

6.4.1. Provisioned D-Channel X.25 Parameters

```
DX25_CB ::= [PRIVATE 89] IMPLICIT SET {

    -- hex value X'ff59'

    -- Number of provisioned permanent virtual circuits
    num_pv                   [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER (0..255),

    -- Number of incoming logical channels
    num_in_lc                [CONTEXT-SPECIFIC 32] IMPLICIT INTEGER (0..255),

    -- Number of two-way logical channels
    num_2way_lc              [CONTEXT-SPECIFIC 33] IMPLICIT INTEGER (0..255),

    -- Number of outgoing logical channels
    num_out_lc               [CONTEXT-SPECIFIC 34] IMPLICIT INTEGER (0..255),

    -- Sequence number modulus
    seq_num_mod              [CONTEXT-SPECIFIC 35] IMPLICIT Seq_num_mod,
    -- CUG index list is for further study
    -- additional parameters are for further study }
```

6.4.2. Provisioned B-Channel X.25 Parameters

```
BX25_CB ::= [PRIVATE 90] IMPLICIT SET {

   -- hex value X'ff5a'

   -- B channel number
   b_ch_num                 [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER (1..30),

   -- Number of permanent virtual circuits
   num_pvc                  [CONTEXT-SPECIFIC 32] IMPLICIT INTEGER (0..255),

   -- Number of incoming logical channels
   num_in_lc                [CONTEXT-SPECIFIC 33] IMPLICIT INTEGER (0..255),

   -- Number of two-way logical channels
   num_2way_lc              [CONTEXT-SPECIFIC 34] IMPLICIT INTEGER (0..255),

   -- Number of outgoing logical channels
   num_out_lc               [CONTEXT-SPECIFIC 35] IMPLICIT INTEGER (0..255),

   -- Sequence number modulus
   seq_num_mod              [CONTEXT-SPECIFIC 36] IMPLICIT Seq_num_mod,

   -- CUG index list is for further study
   -- Additional parameters are for further study }
```

6.4.3. Voice Parameters

```
   -- the inclusion of parameters for the following is for further study
                            -- call hold option
                            -- conference call provisioning
                            -- drop option
                            -- transfer option
                            -- key system services
                            -- supplementary services

VOICE_CB ::= [PRIVATE 91] IMPLICIT SET {

   -- hex value X'ff5b'

   -- Number of call appearances supported
   num_ca                   [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER (0..255),

   -- codec law (u/a law)
   codec_law                [CONTEXT-SPECIFIC 32] IMPLICIT Codec_law,

   -- phone volume (receiver gain)
   -- decimal number of db above lowest gain
   volume                   [CONTEXT-SPECIFIC 33] IMPLICIT INTEGER (0..28)}
```

6.4.4. D-Channel LAPD Parameters

```
D_LAPD_CB ::= [PRIVATE 92] IMPLICIT SET {

   -- hex value X'ff5c'

   -- SAPI-16 TEI. A decimal number 0 to 63 and 255.
   -- The value 255 turns on automatic selection
   sapi16_tei               [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER (0..255),

   -- SAPI-16 parameter negotiation by C.O.
   -- FALSE, if parameter negotiation is disabled
   -- TRUE, if enabled
   sapi16_nego              [CONTEXT-SPECIFIC 32] IMPLICIT BOOLEAN,

   -- SAPI-0 TEI. A decimal number 0 to 63 and 255.
   -- The value 255 turns on automatic selection
   sapi0_tei                [CONTEXT-SPECIFIC 33] IMPLICIT INTEGER  (0..255)}
```

6.4.5. Basic Rate (BRI) Provisioning Parameters

```
BASIC_CB ::= [PRIVATE 93] IMPLICIT SET {

   -- hex value X'ff5d'

   -- SPID = Service profile identifier
   spid                     [CONTEXT-SPECIFIC 31] IMPLICIT PrintableString;

   -- The value of the following D/B channel capabilities are as follows:
   --    0   none
   --    1   X.25 packet switched (PSD)
   --    2   circuit switched voice (CSV)
   --    4   circuit switched data (CSD)
   --    8   packet on demand (POD)
   --    Even values may be added to indicate the provisioning of a B channel for
   --        several uses.
   --    d_ch_capab values are restricted to 0 or 1.
   d_ch_capab               [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER;
   b1_ch_capab              [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER;
   b2_ch_capab              [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER;

   -- Enblock/overlap sending is for further study
   -- Switch type is for further study
   -- Switch software generic is for further study
   -- Additional parameters are for further study  }
```

6.4.6. Primary Rate (PRI) Provisioning Parameters

```
PRI_CB ::= [PRIVATE 94] IMPLICIT SET {

   -- hex value X'ff5e'

   -- FOR FURTHER STUDY }
```

6.4.7. Vendor Specific Parameters

These parameters are to be provided by the AE/NA vendor in a standard form (to be determined) that can be accessed by the PE or combined with the PE in order to provide NA configuration control by the PE. This is for further study.

```
VENDOR_CB ::= [PRIVATE 95] IMPLICIT SET {

   -- hex value X'ff5f'

   -- FOR FURTHER STUDY }
```

Possible parameters are:

- network adaptor attributes
- register recall timing
- switch attributes that affect NA behavior
- microcode

6.5. General Data Blocks

In addition to the ASI commands associated with the Session and Configuration data blocks some other ASI commands, but not all, have associated data blocks. This section defines the data blocks for those ASI commands that have associated data blocks, other ASI commands have a Length Parameter of zero.

This section is organized into two sub-subsections which describe commands associated with the management and control planes.

6.5.1. Management Plane Commands

6.5.1.1. Nb-AE_STATUS confirmation Data Block

This data block conveys information about the status/state of the AE. Its contents are for further study. Possible parameters are:

- A list of all NAs (list of Adaptor_id primitive types).
- Identification of the AE.
- Vendor, version and revision.
- Active channels. A list of channels that are connected.
- A profile of the current AE resource utilization.
- Map of call appearances and associated sessions (session_id).
- Result of most recent AE sanity diagnostic.

- Information related to supplementary services.

6.5.1.2. Nb-CAPABILITY confirmation Data Block

This confirmation provides a listing of the AE Capabilities in response to a Nb-CAPABILITY request.

The completion of this section is for further study.

```
-- Types of channels supporting services (D, B1, B2, and/or H channels)
-- Types of CSD supported and associated channel (clear/B1, v.120/B2, other)
-- Types of PSD supported and associated channel(provisioned/B1, POD_in/B2,
-- POD_out/B2, Dch...)
-- Available RAM (in some basic unit that is processor independent)
-- Simultaneous capabilities (CSV+CSD, CSD+CSD, CSD+CSD+DPKT...)
-- Others are for further study
```

6.5.1.3. Nb-CONFIGURE confirmation Data Block

```
status                   [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER {
                         success(0),
                         failure(1)}
```

6.5.1.4. Nb-NA_INFO request Data Block

This command is used to request Network Adaptor specific information which is returned by the Nb-NA_INFO confirmation command. The Adaptor ID parameter is used to select a specific NA. When the Adaptor ID has a value of 0xFFFF, all NAs are selected and multiple Nb-NA_INFO confirmations are generated.

```
adaptor_id               adaptor_id,
```

6.5.1.5. Nb-NA_INFO confirmation Data Block

This command reports information about a specific NA. If the Nb-INFO request asks for all of them, by requesting an Adaptor ID of 0xFFFF, each NA will be reported in a separate confirmation. A NA cannot have an Adaptor ID of 0xFFFF.

```
adaptor ID               Adaptor_id
na_serial_num            [CONTEXT-SPECIFIC 31] IMPLICIT PrintableString,
vendor                   [CONTEXT-SPECIFIC 32] IMPLICIT PrintableString,
firmware_ver             [CONTEXT-SPECIFIC 33] IMPLICIT PrintableString,
hardware_ver             [CONTEXT-SPECIFIC 34] IMPLICIT PrintableString,

-- True = All Tests Passed
self_test                [CONTEXT-SPECIFIC 35] IMPLICIT BOOLEAN,

-- memory_utilization is for further study.
-- Self Test results, including hardware errors such as:
--       memory parity errors, bus errors, timeouts and signaling
--       errors are for further study.
-- Hook State is for further study.
-- additional parameters are for further study
```

6.5.1.6. Nb-RESET confirmation Data Block

```
 -- the parameter module is for further study.
module                   [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER SIZE (2),
result                   [CONTEXT-SPECIFIC 32] IMPLICIT INTEGER {
                         success (0),
                         failure (1)},
```

6.5.1.7. Nb-RESET request Data Block

```
--  the parameter module is for further study.
module                   [CONTEXT-SPECIFIC 31] IMPLICIT INTEGER SIZE (2),

-- All values of r_mode except r_mode = 0 are for further study
-- Refer to Chapter 5 for values of reset mode.
r_mode                   [CONTEXT-SPECIFIC 32] IMPLICIT INTEGER SIZE (2),
```

6.5.2. Call Control Plane Commands

6.5.2.1. Nb-CONNECT_STATUS confirmation Data Block

```
channel_in_use           Channel,
associated_sessions      -- list of associated session_ids.
lcn                      -- logical channel number or some indication of none.
own_use                  -- owner or user session.
                         -- If associated_sessions = none the value is a don't
                            care.
call_state               [CONTEXT-SPECIFIC 31] IMPLICIT Call_state,
other                    -- for further study
```

6.5.2.2. Nb-DISCONNECT request Data Block and Nb-DISCONNECT response Data Block

```
cause                    Cause,
local_dir_num            [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number,
local_sub_addr           [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress, *

-- The following X.25 parameters are for further study.  Only one of
-- the x.25_*_cause and cause (ISDN) can be used a given message.
x25_clearing_cause       X25_clearing_cause
x25_reset_cause          X25_reset_cause
x25_restart_cause        X25_restart_cause
x25_diag                 X25_diag

-- Additional parameters are for further study
```

6.5.2.3. Nb-DISCONNECT indication Data Block

```
cause                     Cause,
distant_dir_num           [CONTEXT-SPECIFIC 31] IMPLICIT Directory_number,
distant_sub_addr          [CONTEXT-SPECIFIC 32] IMPLICIT Subaddress,

-- The following X.25 parameters are for further study.  Only one of
-- the x.25_*_cause or cause (ISDN) can be used in one message.
x25_clearing_cause        X25_clearing_cause
x25_reset_cause           X25_reset_cause
x25_restart_cause         X25_restart_cause
x25_diag                  X25_diag

-- Additional parameters are for further study
```

6.5.2.4. Nb-DISCONNECT confirmation Data Block

```
-- only for x.25 cause and to carry x.25 user data

-- The following X.25 parameters are for further study.  Only one of
-- the x.25_*_cause or cause (ISDN) can be used in one message.

x.25_reset_cause          X.25_reset_cause
x.25_restart_cause        X.25_restart_cause
x.25_diag                 X.25_diag

-- Additional parameters are for further study
```

6.5.2.5. Nb-ERROR indication Data Block

```
class                     [CONTEXT-SPECIFIC 31] IMPLICIT Error_class,
code                      [CONTEXT-SPECIFIC 32] IMPLICIT OCTECT
                          {unknown (0) -- not one of the following

                          -- resource errors
                          unknown_res_type (1),
                          res_not_accepting_requests (2),
                          no_res_available (3),
                          res_type_not_configured (4),

                          -- syntax errors
                          invalid_command (11),
                          invalid_command_length (12),
                          invalid_parameter_tag(13),
                          invalid_parameter_length (14),
                          invalid_parameter_value (15),
                          invalid_aei (16),
                          invalid_pei (17),
                          invalid_aei_pei_pair (18),
                          insufficient_parameters (19),

                          -- state errors
                          not_valid_connect_phase (31),
                          not_valid_disconnect_phase (32),
                          not_valid_active_phase (33),
                          not_valid_idle_phase (34)}

-- indicates the tag of an associated errored parameter
parameter_tag             [CONTEXT-SPECIFIC 33] IMPLICIT OCTECT STRING SIZE (2),
command_name              [CONTEXT-SPECIFIC 34] IMPLICIT OCTECT STRING SIZE (2),
-- additional parameters are for further study
```

6.5.2.6. Nb-EVENT request Data Block

```
event_type                [CONTEXT-SPECIFIC 31] IMPLICIT Event_type,

-- event indications
progress_ind              [CONTEXT-SPECIFIC 33] IMPLICIT Progress_ind,
cause                     [CONTEXT-SPECIFIC 35] IMPLICIT Cause,

-- The inclusion of x.25 parameters is for further study
```

6.5.2.7. Nb-EVENT indication Data Block

```
event_type                [CONTEXT-SPECIFIC 31] IMPLICIT Event_type,

-- event indications
display                   [CONTEXT-SPECIFIC 32] IMPLICIT PrintableString,
progress_ind              [CONTEXT-SPECIFIC 33] IMPLICIT Progress_ind,
signal                    [CONTEXT-SPECIFIC 34] IMPLICIT INTEGER (0..127),
cause                     Cause,
```

6.5.2.8. Nb-MORE_INFO response Data Block

```
   digits                    [CONTEXT-SPECIFIC 31] IMPLICIT PrintableString,
```

6.5.2.9. Nb-USER_USER_DATA indication Data Block

```
   user_to_user              User_to_user,
```

6.5.2.10. Nb-USER_USER_DATA request Data Block

```
   user_to_user              User_to_user,
```

6.6. Data Block Coding

This section provides typical examples of ASI encoding as an aid in understanding the conversion, from the notation used in the rest of this chapter, to the ASI code. In some of the following examples optional parameters have been omitted and subaddress which is for further study is used.

6.6.1. Primitive Data Types

6.6.1.1. Address Digits (Addr_digits)

Field	Tag,Length,Contents	Comments
addr_digits	df1f, 07, "5446444"	544-6444

6.6.1.2. Cause Code (Cause)

Field	Tag,Length,Contents	Comments
cause	df23, 01, 01	unassigned number (ISDN cause)

6.6.2. ASI Commands

6.6.2.1. Nb-DISCONNECT request or response

```
Field                        Tag, Length, Contents   Comments

cause                        df23, 01, 01            unassigned number (ISDN cause)
local_dir_num                bf1f, 15
   addr_type                 df20, 01, 20            national
   numbering_plan            df27, 01, 01            E.164
   addr_digits               df1f, 0a, "9085446444"
local_sub_addr               bf20, 0b                for further study
   addr_type                 df2e, 01, 00            NSAP
   addr_info                 df32, 04, "6444"
```

6.6.2.2. Nb-ERROR indication

```
Field                        Tag,Length,Contents     Comments
class                        9f1f, 01, 01            syntax error
code                         9f20, 01, 0f            invalid parameter value
parameter_tag                9f21, 02, bf1f          local directory number
command_name                 9f22, 02, 0104          Nb-CONNECT response
```

6.6.2.3. Nb-CONNECT request with a VOICE SESSION BLOCK

In the following, note that the optional parameter, distant_sub_addr, l is not included.

```
Field                        Tag,Length,Contents     Comments

block_type                   df21, 01, 01            VOICE
local_dir_num                bf1f, 15
   addr_type                 df20, 01, 20            national
   numbering_plan            df27, 01, 01            E.164
   addr_digits               df1f, 0a, "9085446444"
local_sub_addr               bf20, 0b                for further study
   addr_type                 df2e, 01, 00            NSAP
   addr_info                 df32, 04, "6444"
distant_dir_num              bf21, 15
   addr_type                 df20, 01, 02            national
   numbering_plan            df27, 01, 01            E.164
   addr_digits               df1f, 0a, "9142875593"
xfer_cap                     df2d, 01, 00            speech
channel_prefer               ff4e, 08
   channel_type              df24, 01, 01            B channel
   channel_num               9f1f, 01, 01            B1
call_appearance              9f23, 01, 01
user_to_user                 df7f, 0b, “ASI ISSUE 1” OPTIONAL USER TO USER INFORMATION
```

6.6.3. Nb-CONNECT request for a CSD, CLEAR CHANNEL, SESSION

This Session Block is for further study.

```
Field                        Tag,Length,Contents     Comments

block_type                   df21, 01, 03
local_dir_num                bf1f, 15
   addr_type                 df20, 01, 20            national
   numbering_plan            df27, 01, 01            E.164
   addr_digits               df1f, 0a, "9085446444"
local_sub_addr               bf20, 0b                for further study
   addr_type                 df2e, 01, 00            NSAP
   addr_info                 df32, 04, "6444"
distant_dir_num              bf21, 15
   addr_type                 df20, 01, 02            national
   numbering_plan            df27, 01, 01            E.164
   addr_digits               df1f, 0a, "9142875593"
xfer_cap                     df2d, 01, 08            unrestricted data (08)
channel_prefer               ff4e, 08
   channel_type              df24, 01, 01            B channel
   channel_num               9f1f, 01, 02            B2
```

7.0. Formal Description

No formal description of the ASI is available at this time.

The intent is that a formal description in both SDL and Estelle be included in this section at a future date.

8.0. Testing

No testing procedures for the ASI are available at this time.

Appendix A: Call Scenarios

This section provides call scenarios based on the ANS T1.607 SDLs. The purpose of this section is to illustrate the relationship between T1.607 messages and ASI commands. Call scenarios are illustrated based on circuit switched voice.

This appendix is divided into four sections: Outgoing Calls, Incoming Calls, Call Clearing, and Miscellaneous. Each section provides possible scenarios for that call type.

A.1. Circuit Switched Voice

Circuit switched voice calls are defined here as being those voice calls which use the standard 3.1 kHz bearer capability.

A.1.1. Outgoing Calls

Figures A1 - A1.4 provide the various types of outgoing calls. Figures A1 and A1.1 provide the two scenarios where the call setup message is complete and a call is completed. The difference is the optional **ALERTING** generated by the far end in figure A1.1.

Figures A1.2 and A1.3 provide scenarios where the setup message is not complete and additional information is required to complete the call. (Overlap sending mode is indicated and it is assumed that the optional information would appear in the form of the keypad facility). Again, the difference is the optional **ALERTING** generated by the far end.

Figure A1.4 illustrates a call which has been released by the network or far end.

A.1.2. A-1.2 Incoming Calls

Figures A2 - A2.4 provide the various types of incoming calls. Figure A2 illustrates the simplest form of call acceptance: A call arrives and is instantly connected with no optional messages generated.

Figure A2.1 shows an incoming call where the AE sends an **ALERTING** message to signify that the user is being notified (usually, this means the phone is ringing). A **CONNECT** is issued when the user answers.

Figure A2.2 shows an incoming call where the AE sends an **PROCEEDING** message to signify that the call setup is proceeding. A **CONNECT** is issued when the user answers.

Figure A2.3 shows an incoming call where both **PROCEEDING** and **ALERTING** messages are generated.

Figure A2.4 illustrates a call which has been released by the receiving end.

A.1.3. Terminating Calls

Figures A3 - A3.4 illustrate call clearing procedures with either party initiating the call teardown sequence.

Figure A3 has the far end releasing and the near end confirming.

Figure A3.1 has the far end requesting a disconnect and immediately releasing the call. In this case, the near end ignores the disconnect and issues a **RELEASE COMPLETE**.

Figure A3.2 has the far end requesting a disconnect, the near end releasing, and the far end acknowledging the release. Figure A3.3 is the compliment to figure A3.2.

Figure A3.4 has both sides asking for a disconnect at the same time. The near end releases the call and the far end acknowledges.

A.1.4. Miscellaneous

Figures A4 - A4.2 cover the miscellaneous procedures described in T1.607. Figure A4 illustrates the incoming and outgoing **NOTIFY** events.

Figure A4.1 illustrates the incoming and outgoing use of **STATUS ENQUIRY**. Figure A4.2 has an incoming and outgoing **RESTART**. Note that in both A4.1 and A4.2, no messages are passed across the interface.

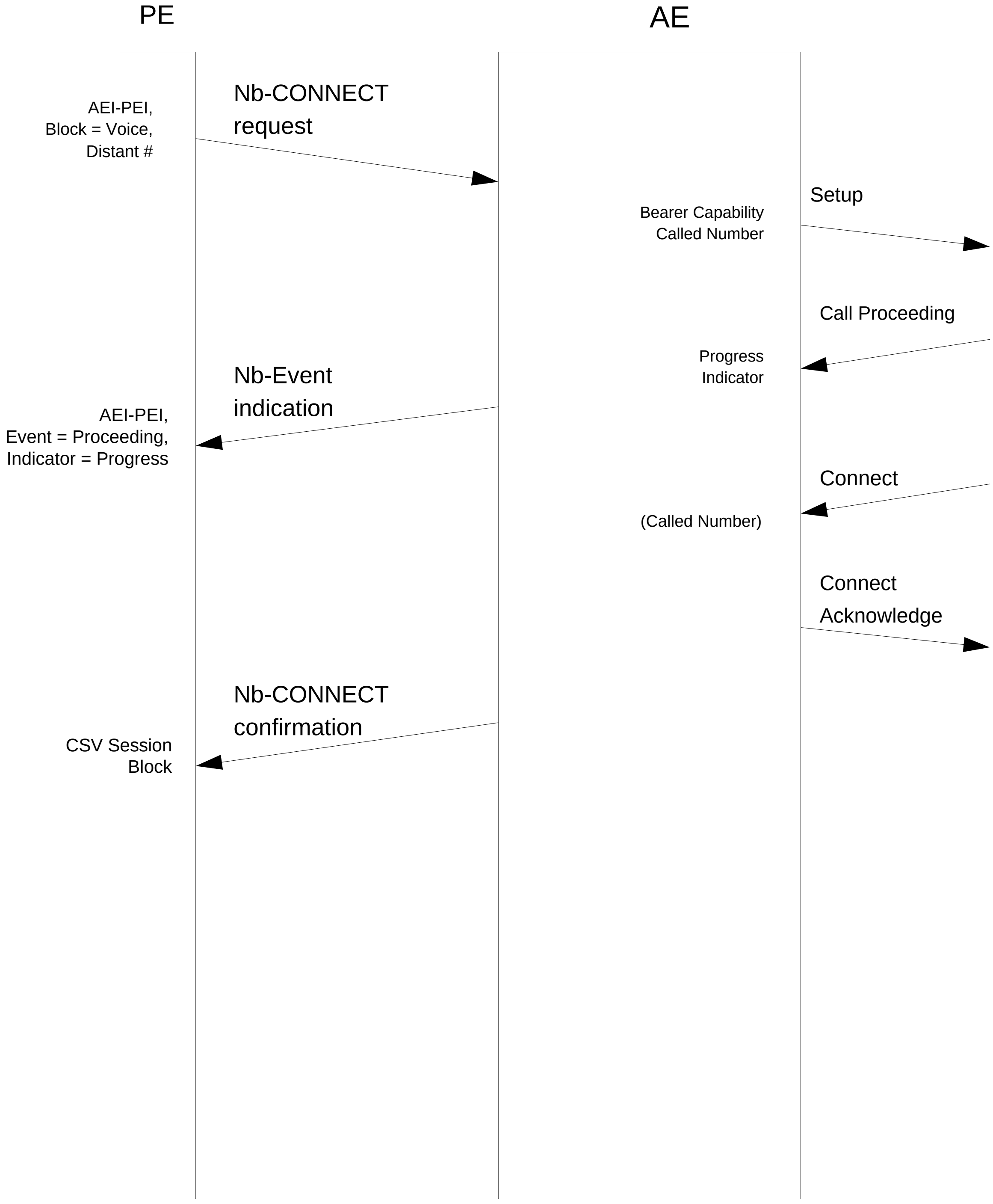


Figure A1

Outgoing call procedure.

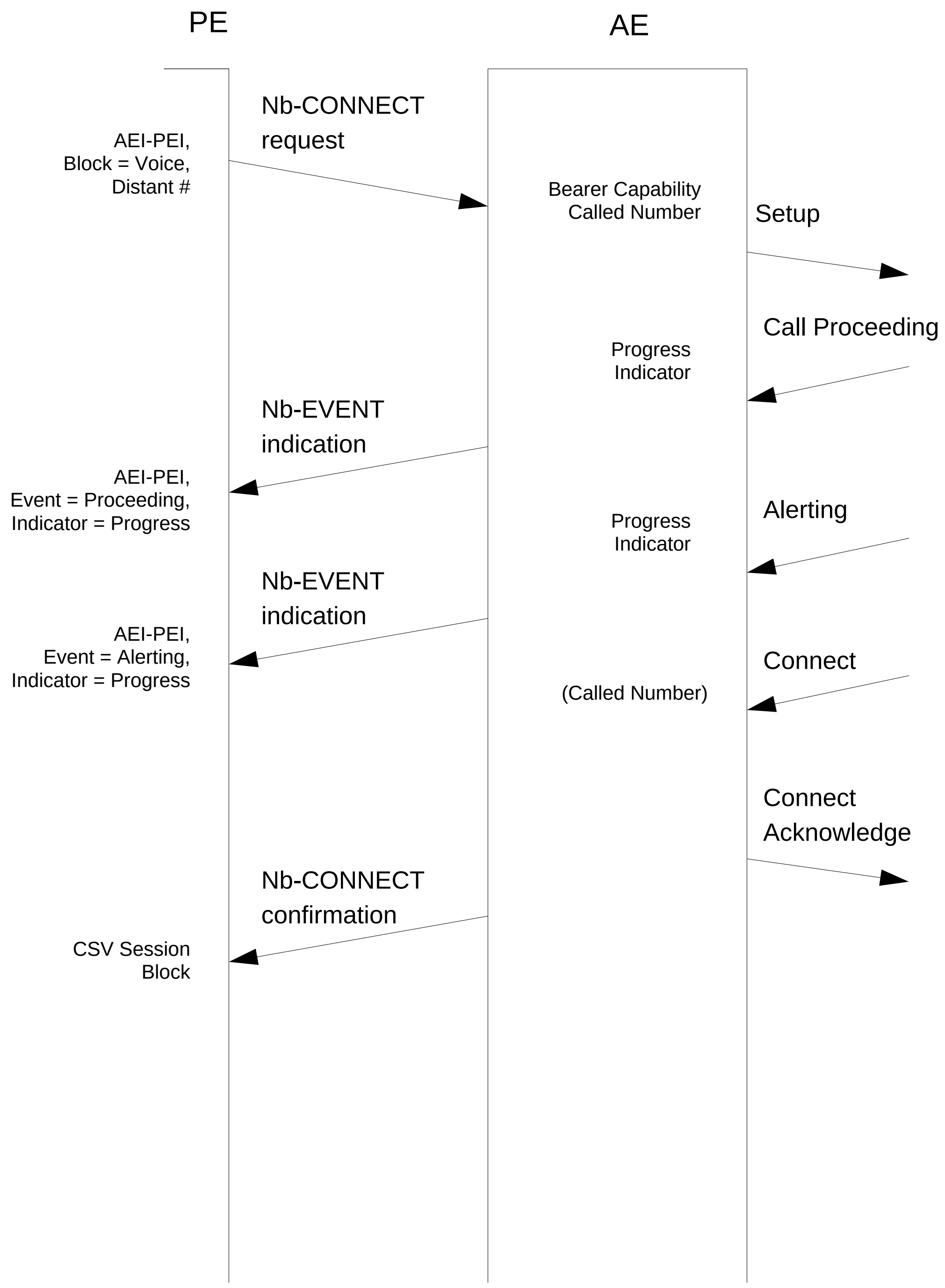


Figure A1.1
Outgoing call procedure.

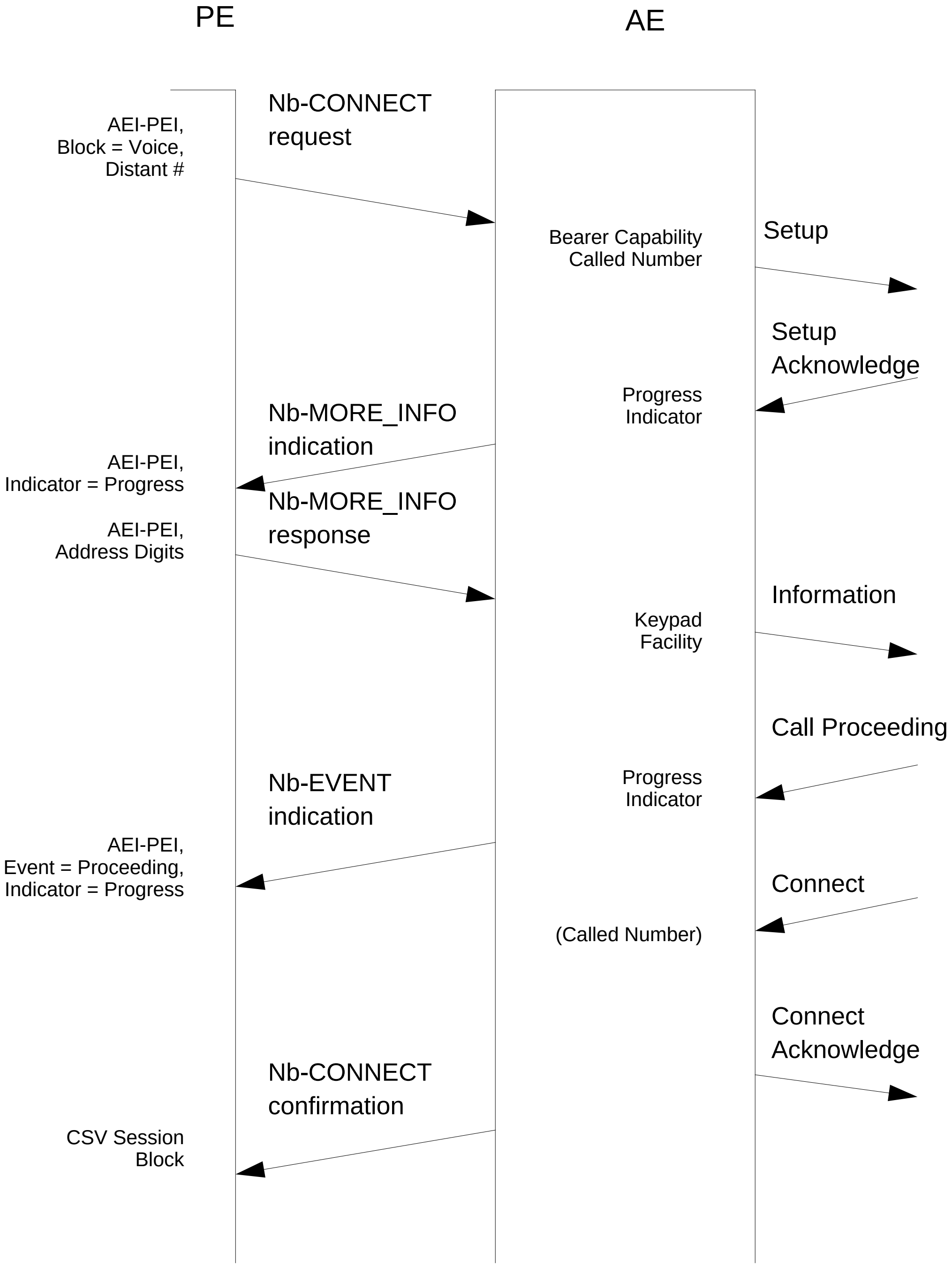


Figure A1.2
Outgoing call procedure.

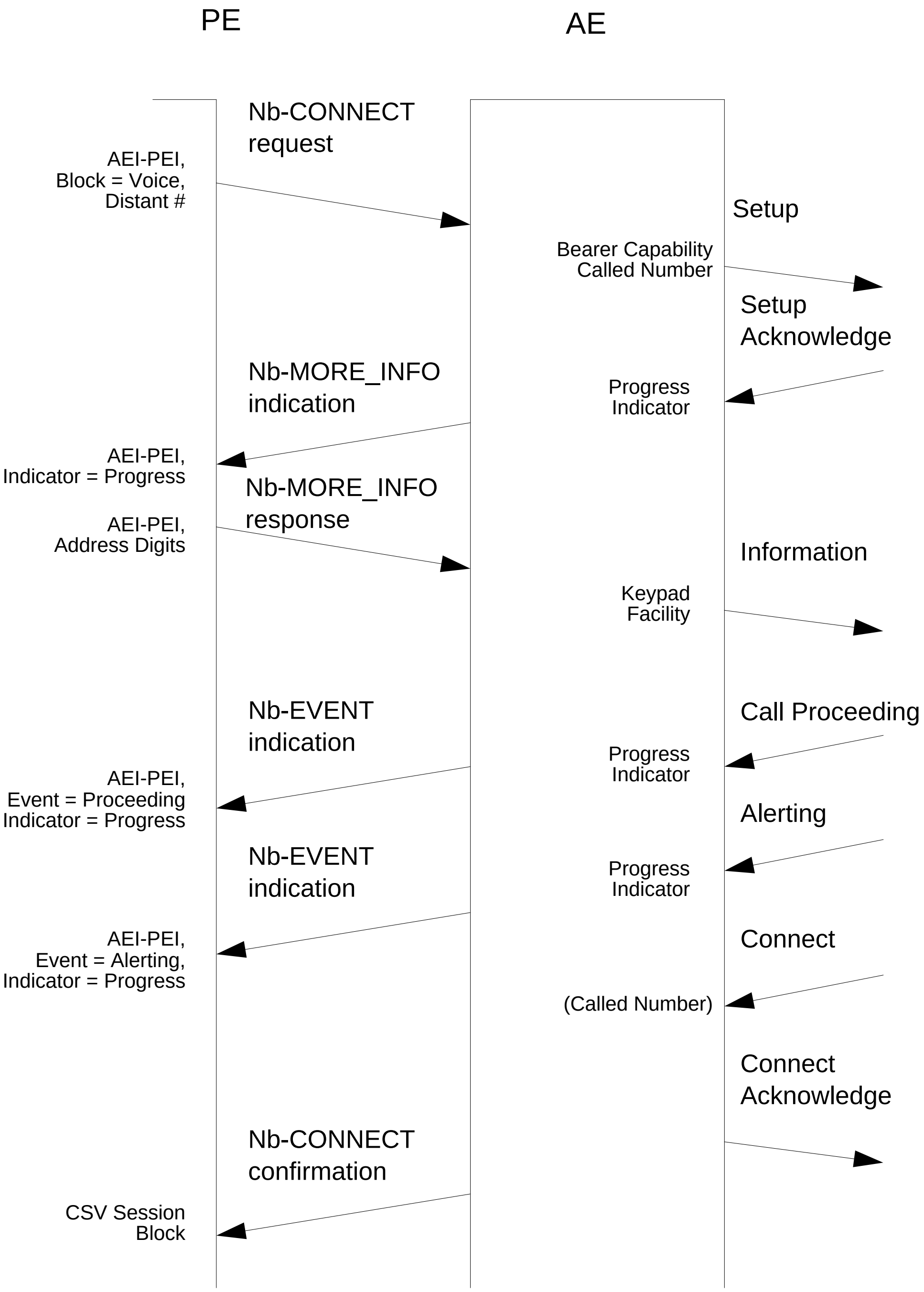


Figure A1.3
Outgoing call procedure.

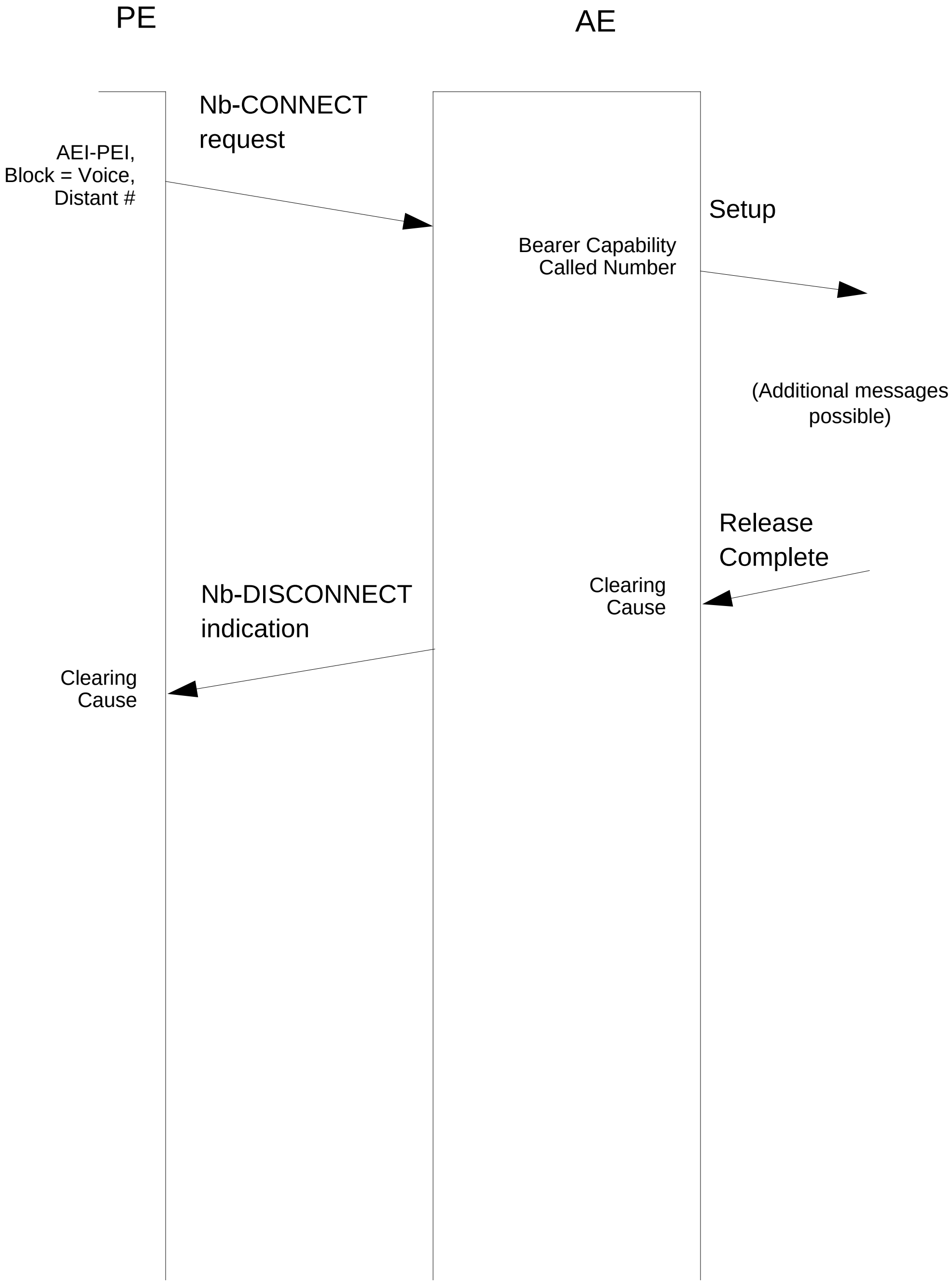


Figure A1.4
Outgoing call procedure.

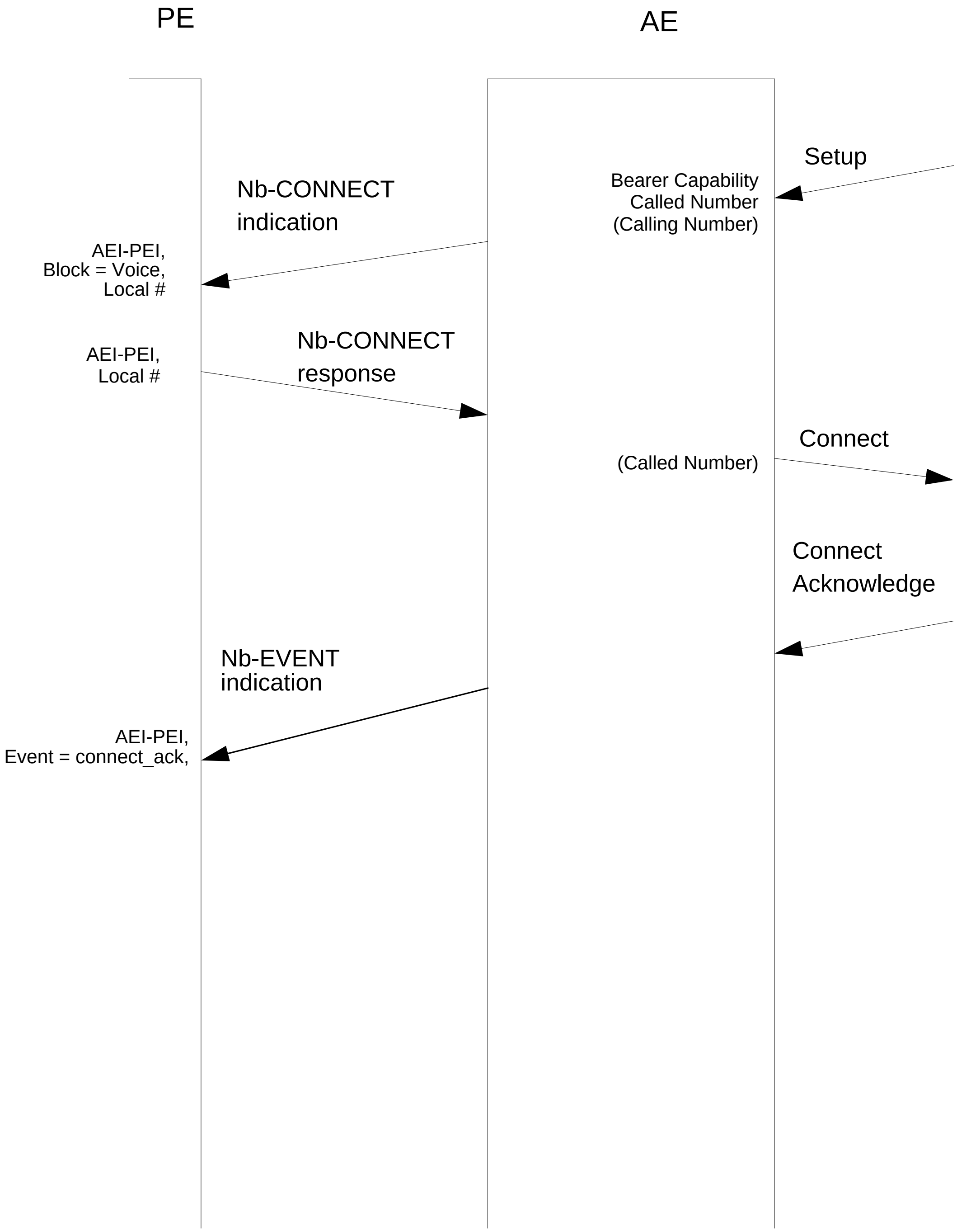


Figure A2
Incoming call procedure.

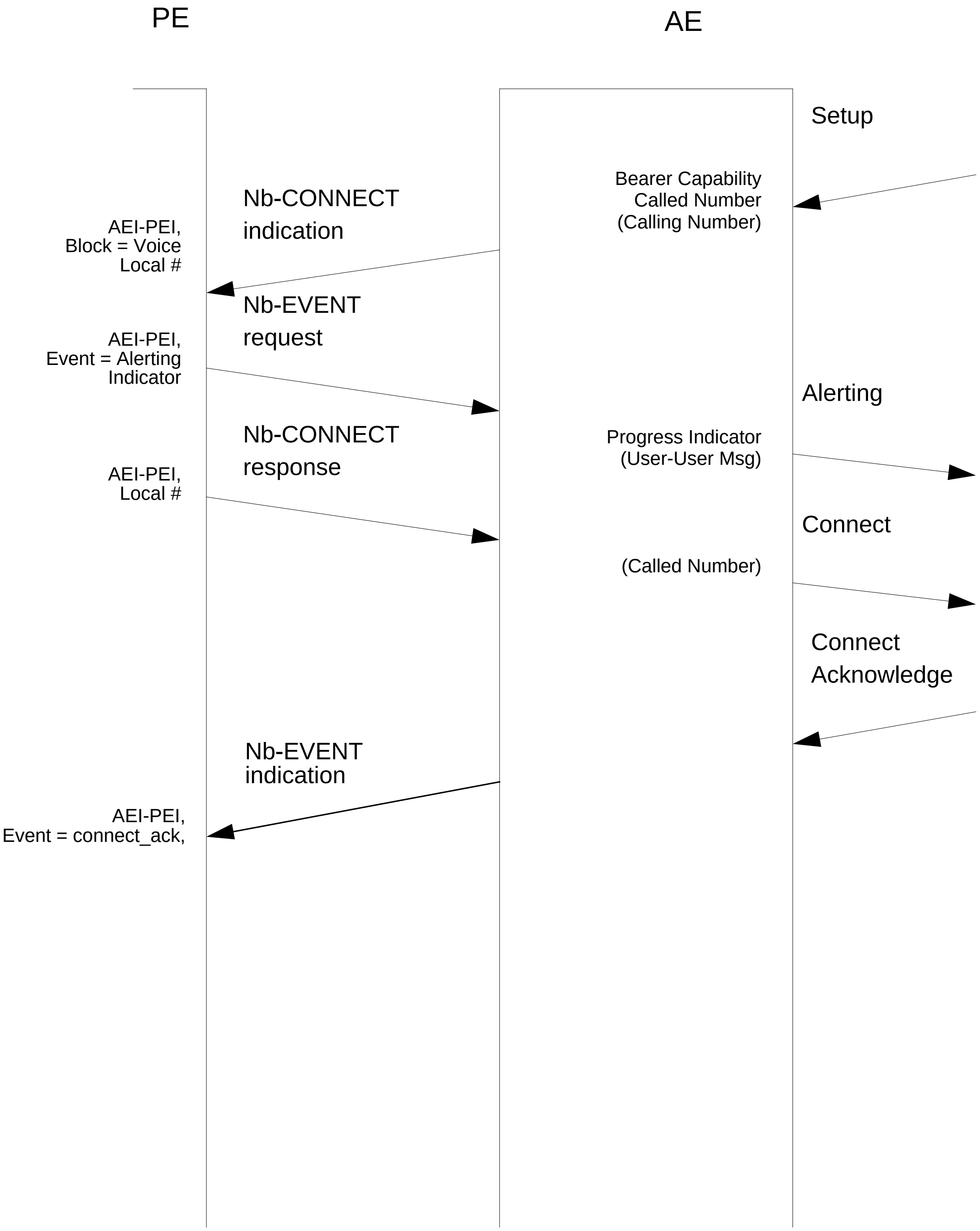


Figure A2.1
Incoming call procedure.

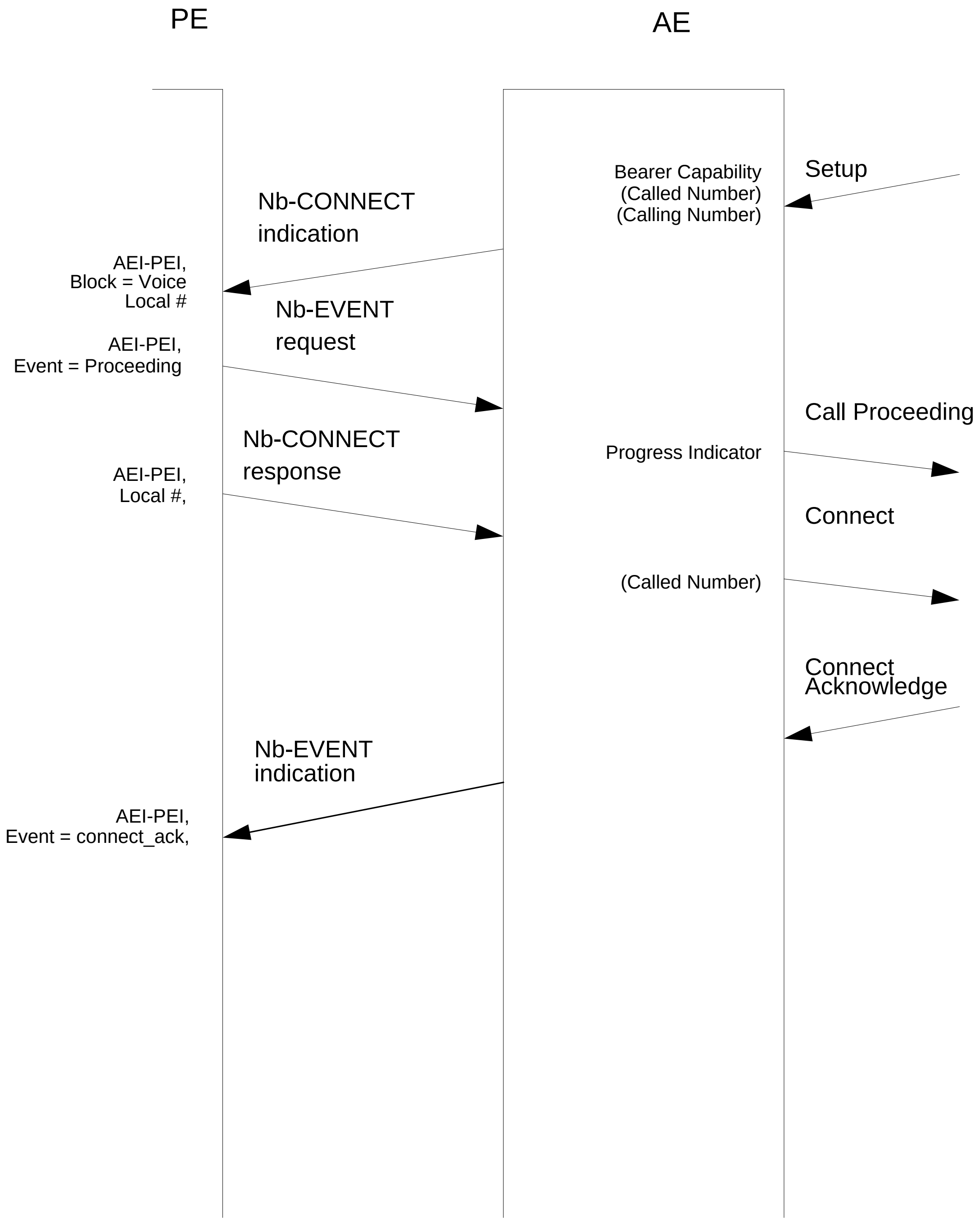


Figure A2.2
Incoming call procedure.

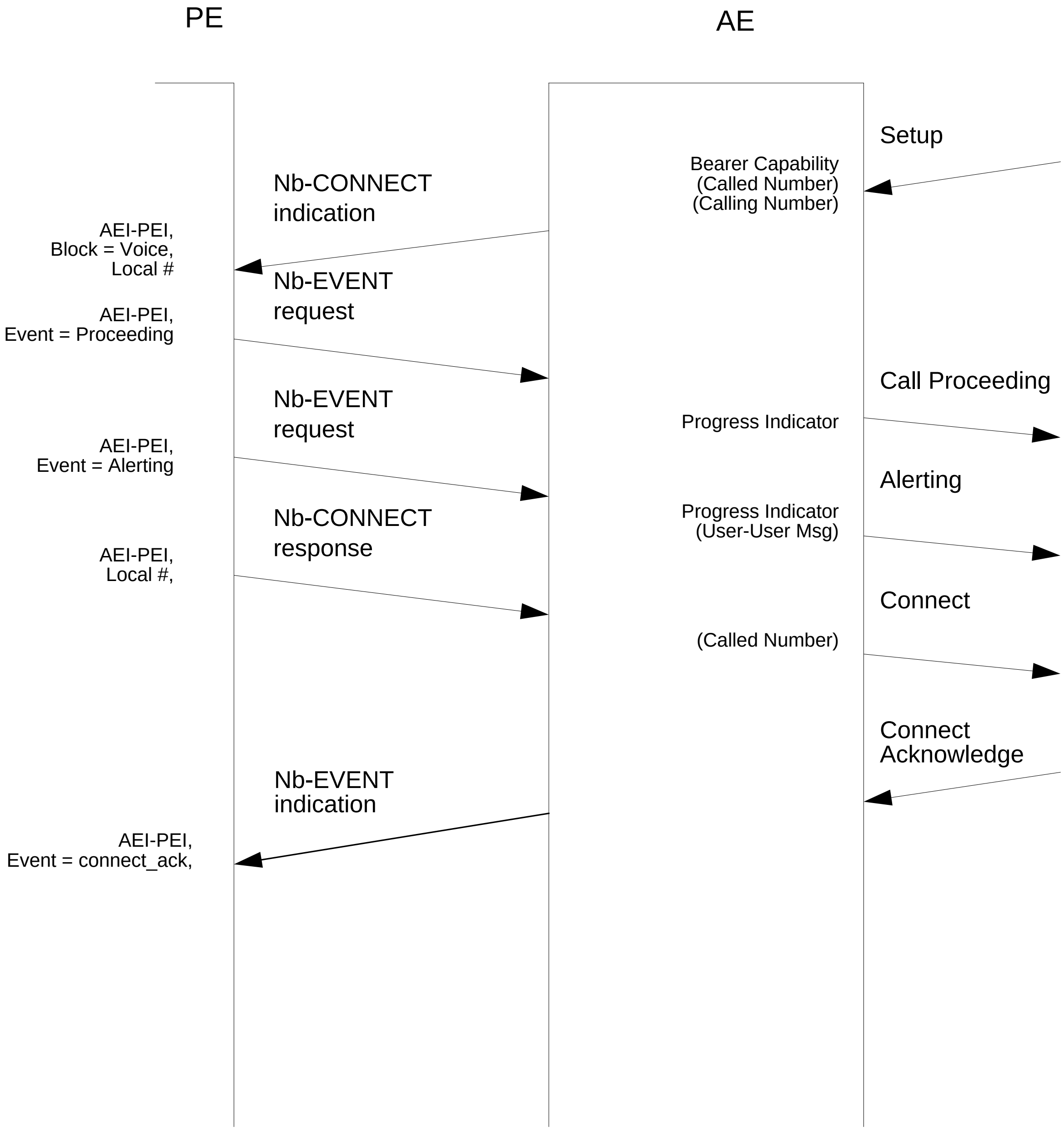


Figure A2.3
Incoming call procedure.

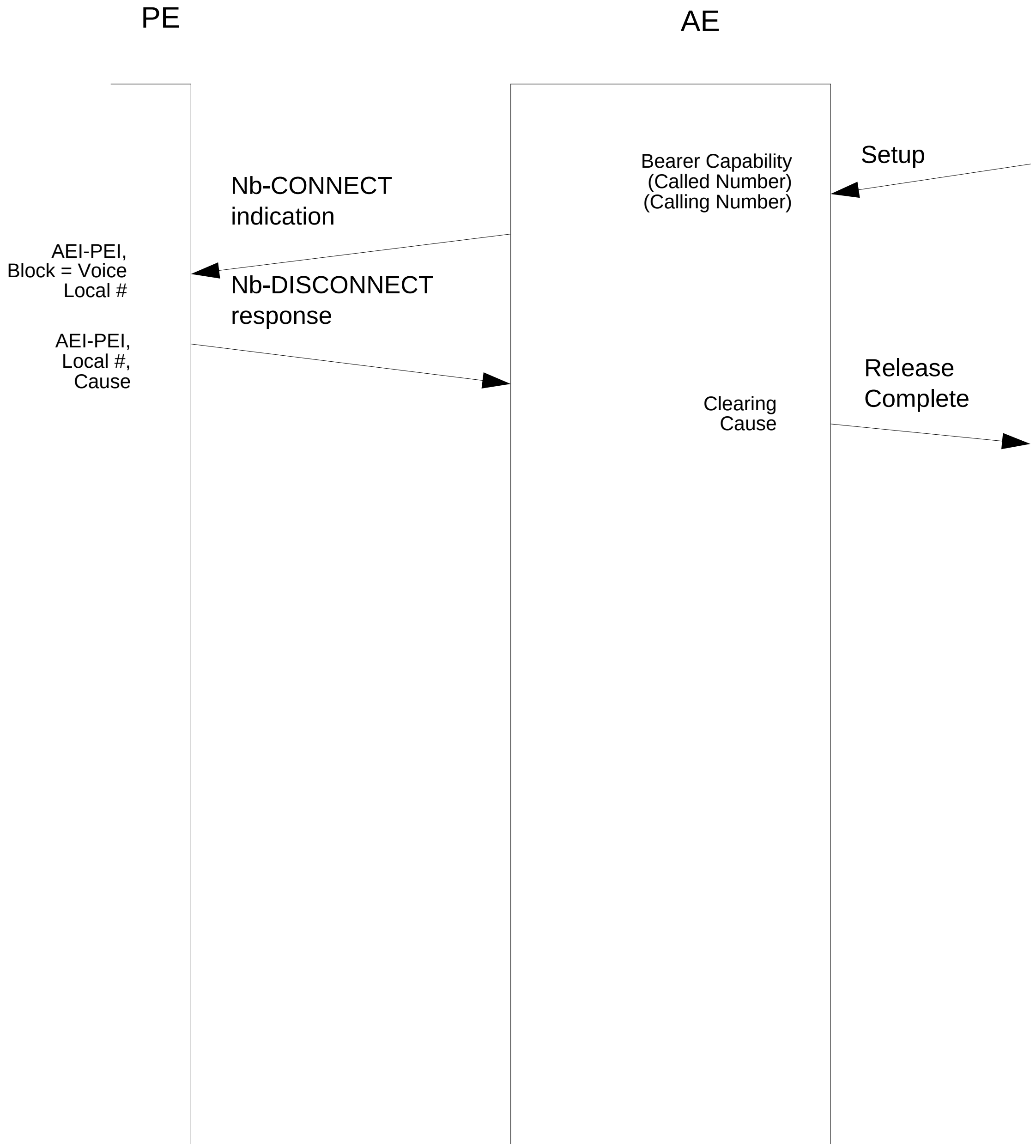


Figure A2.4
Incoming call procedure.

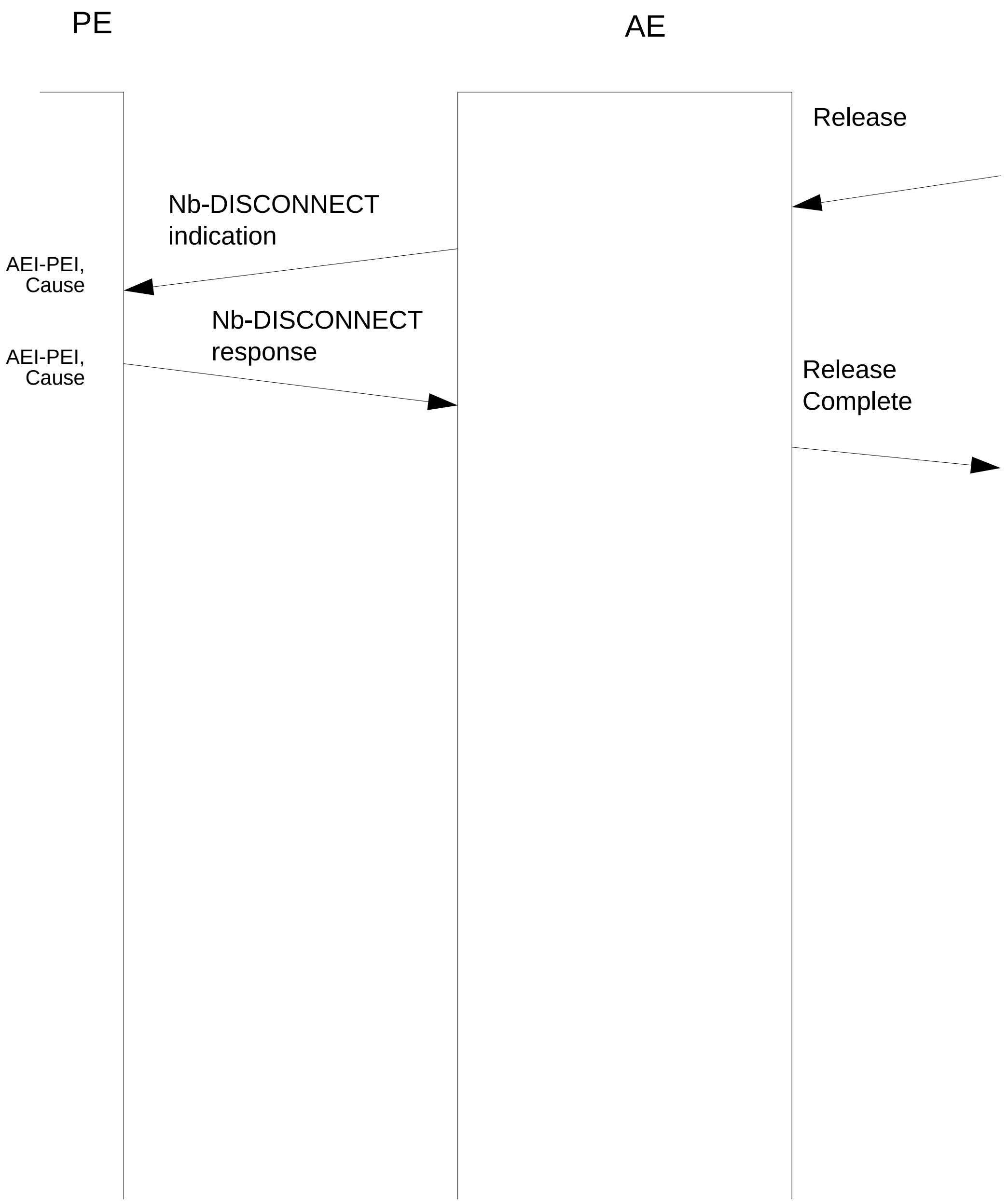


Figure A3
Clearing procedure.

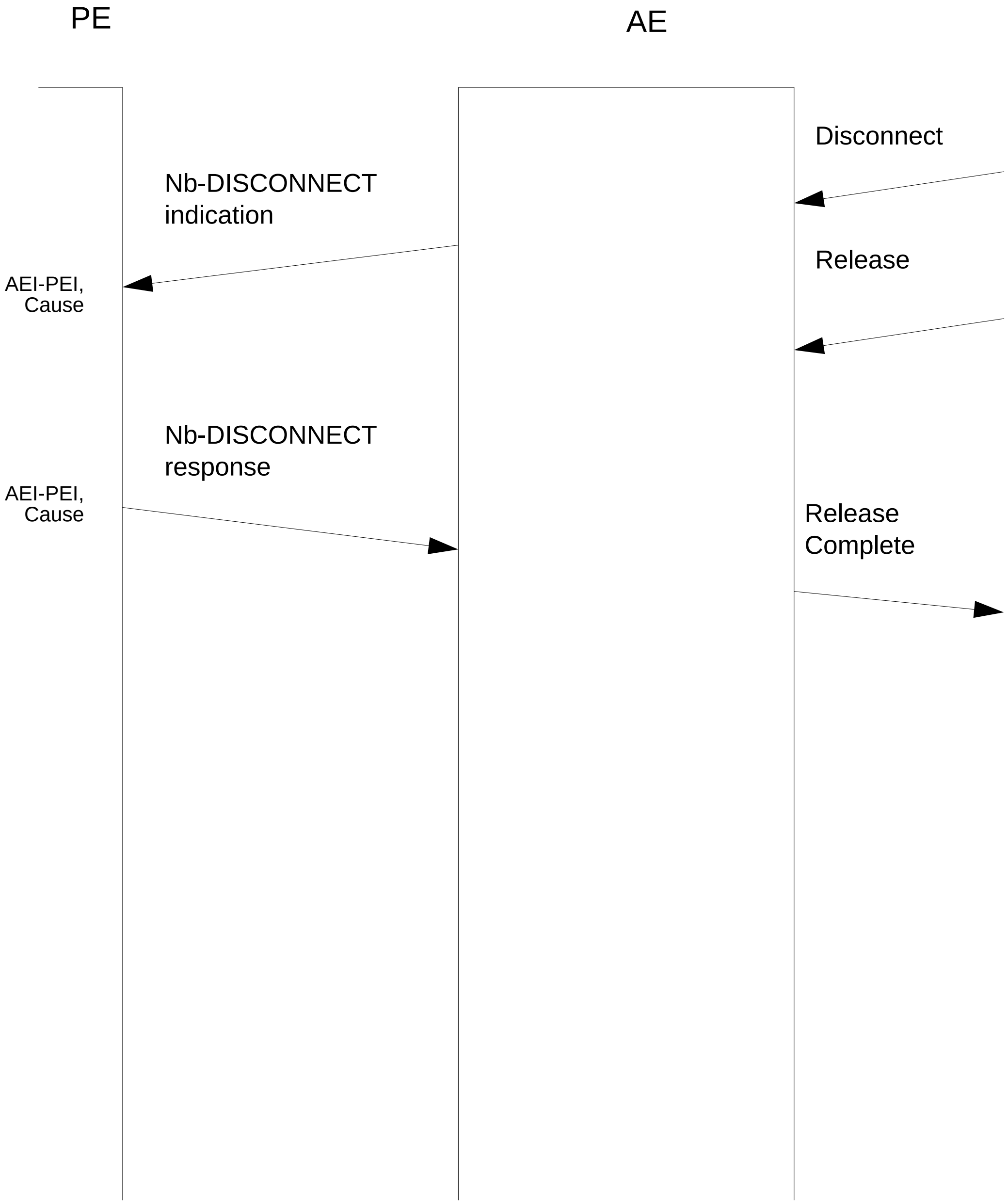


Figure A3.1
Clearing procedure.

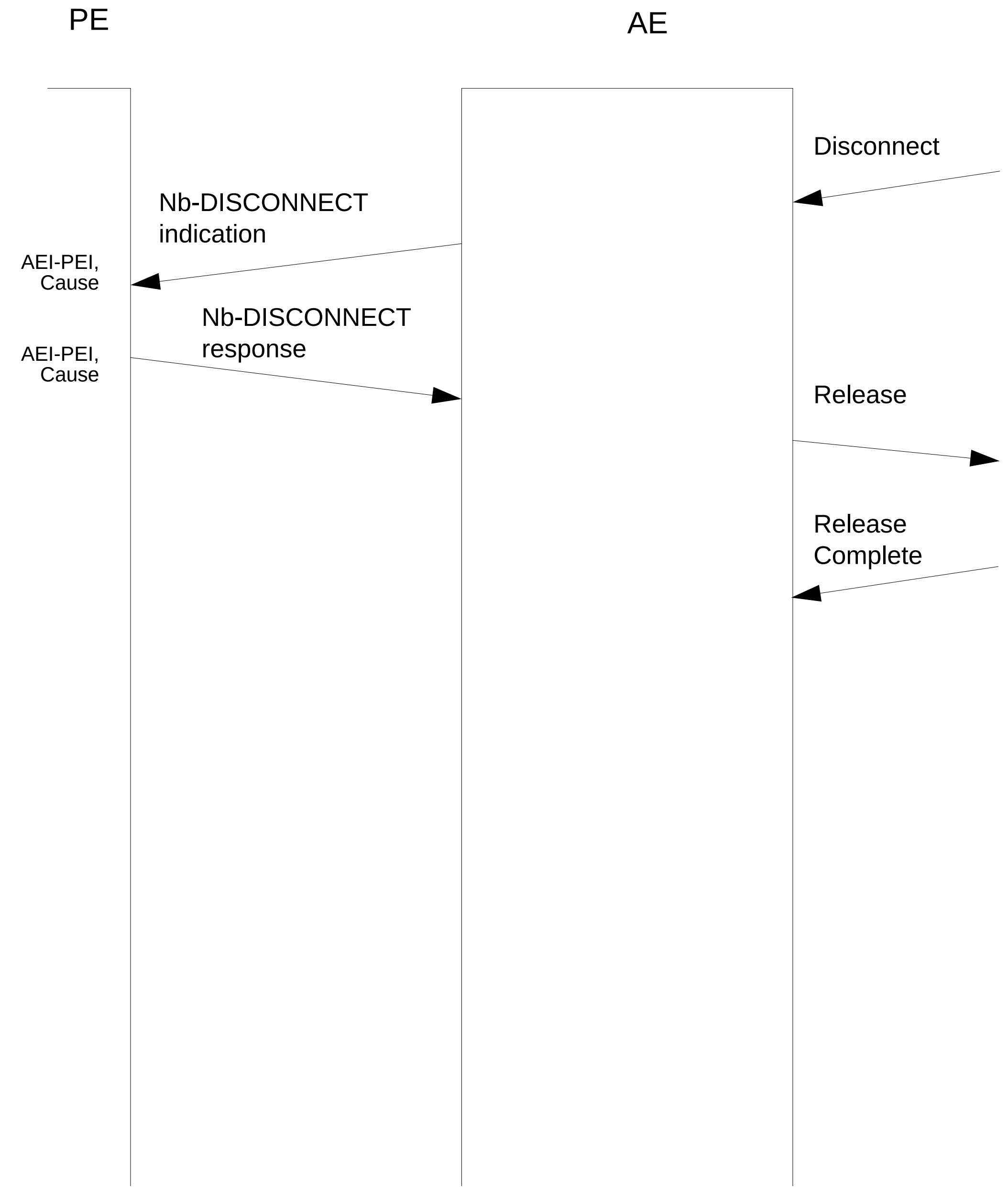


Figure A3.2
Clearing procedure.

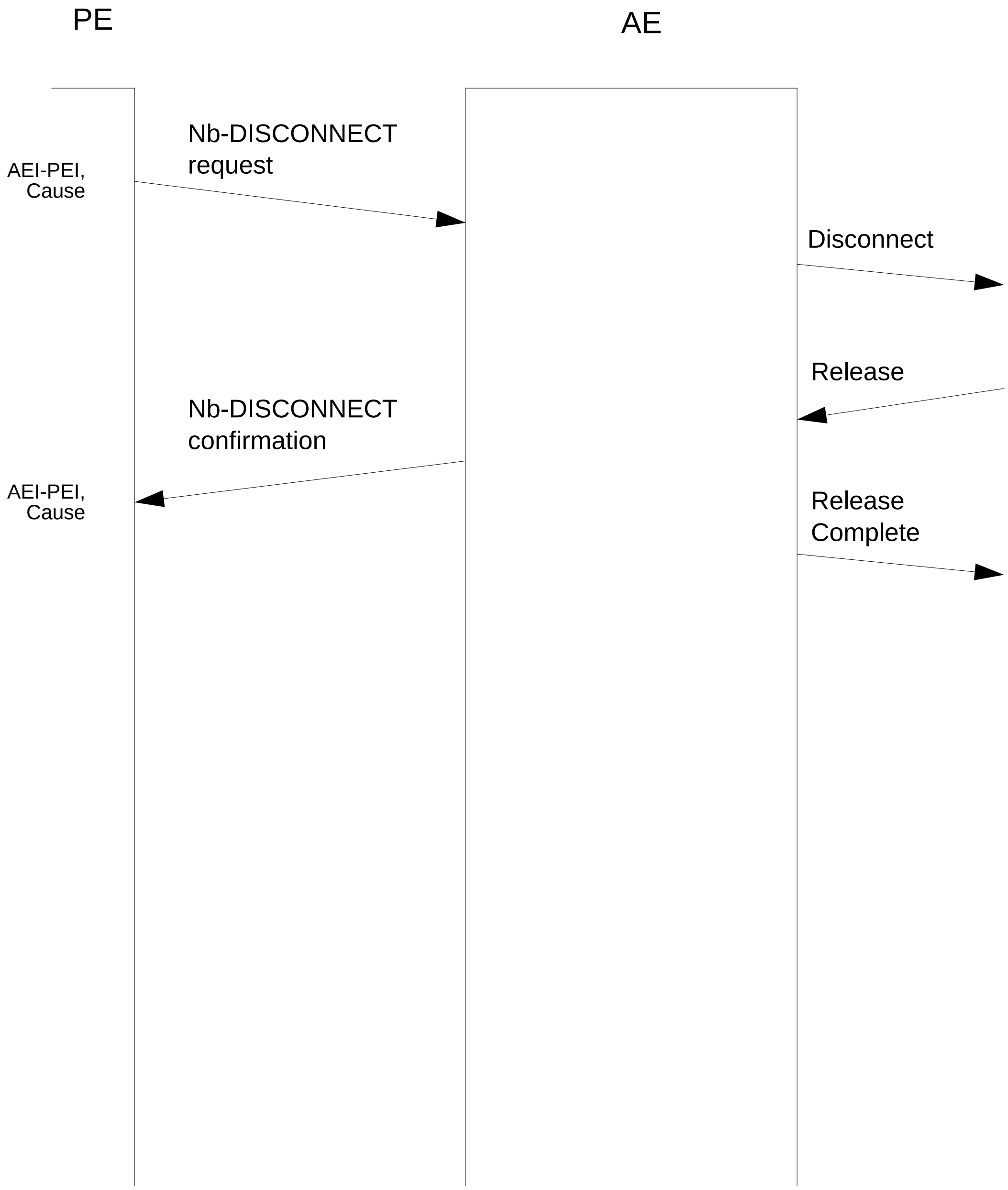


Figure A3.3
Clearing procedure.

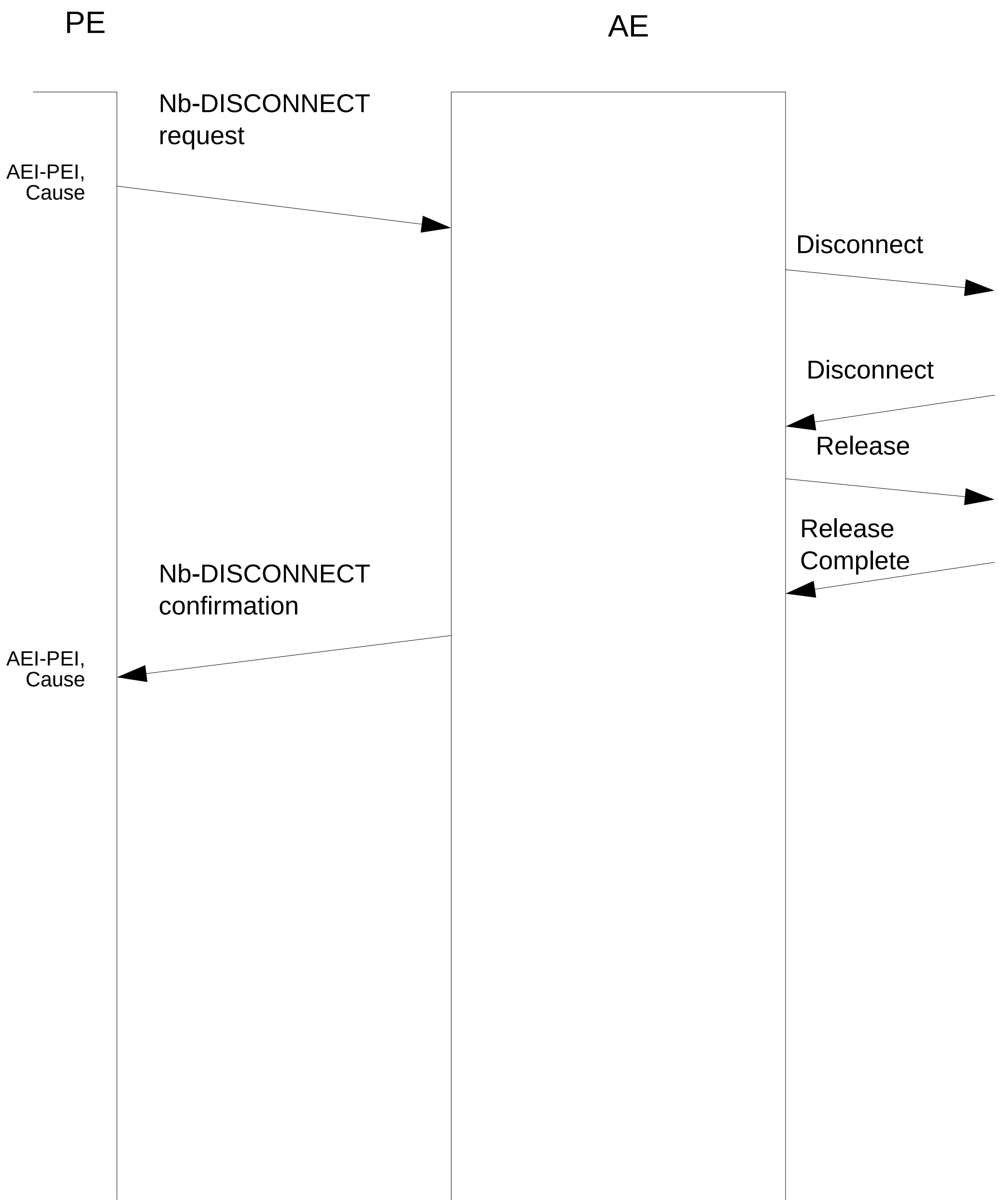


Figure A3.4
Clearing procedure.

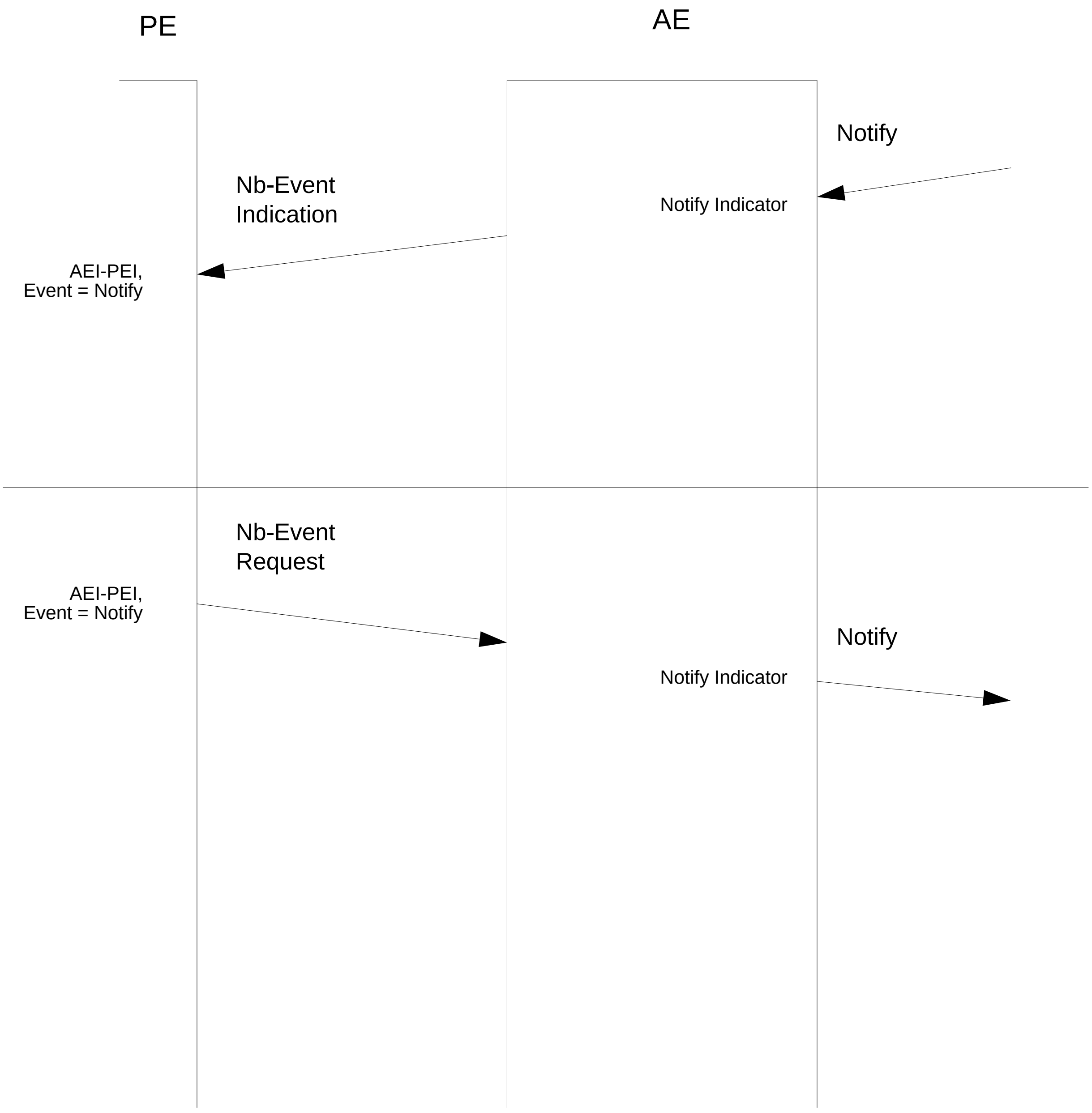


Figure A4
Other procedures.

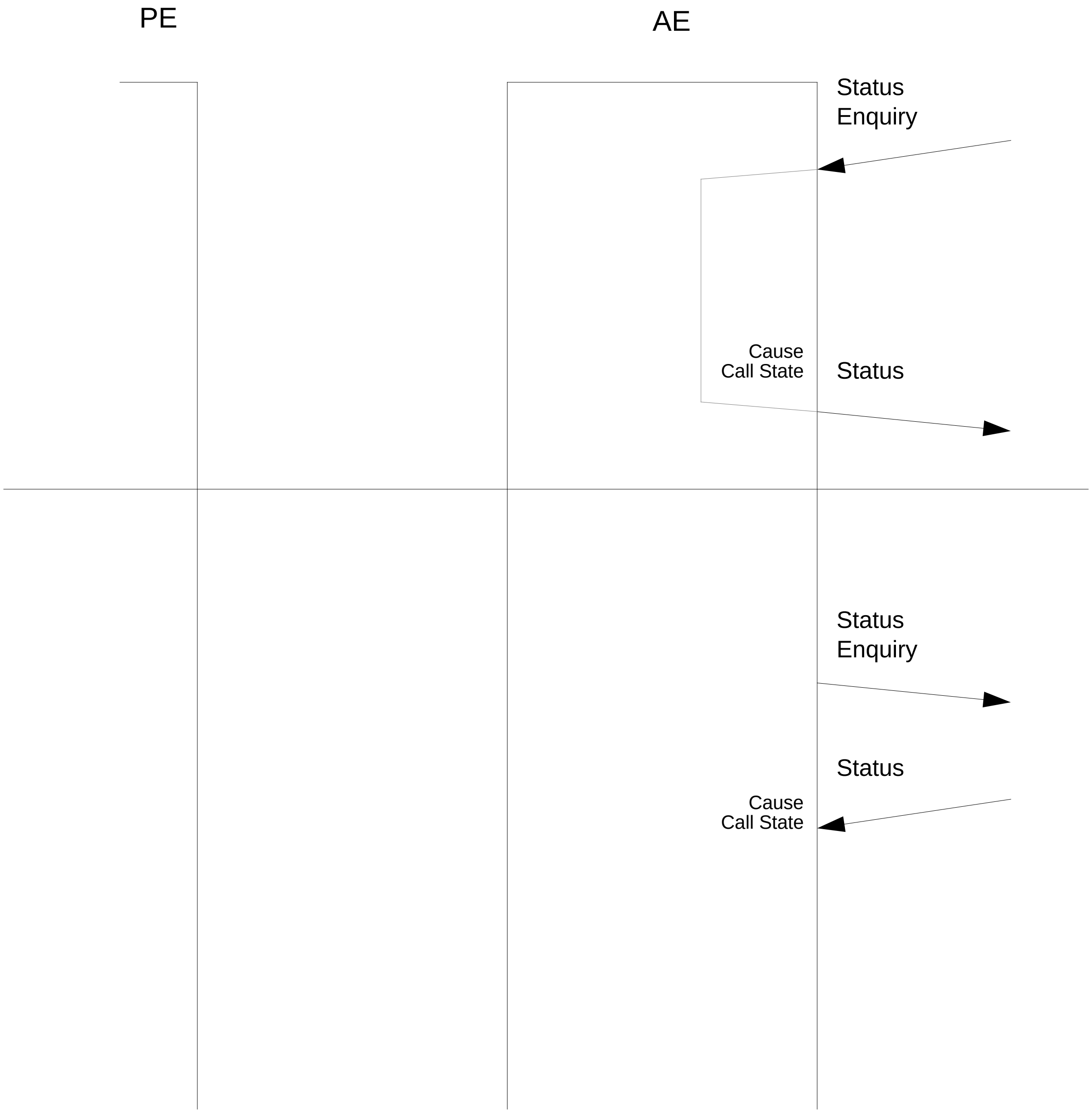


Figure A4.1
Other procedures.

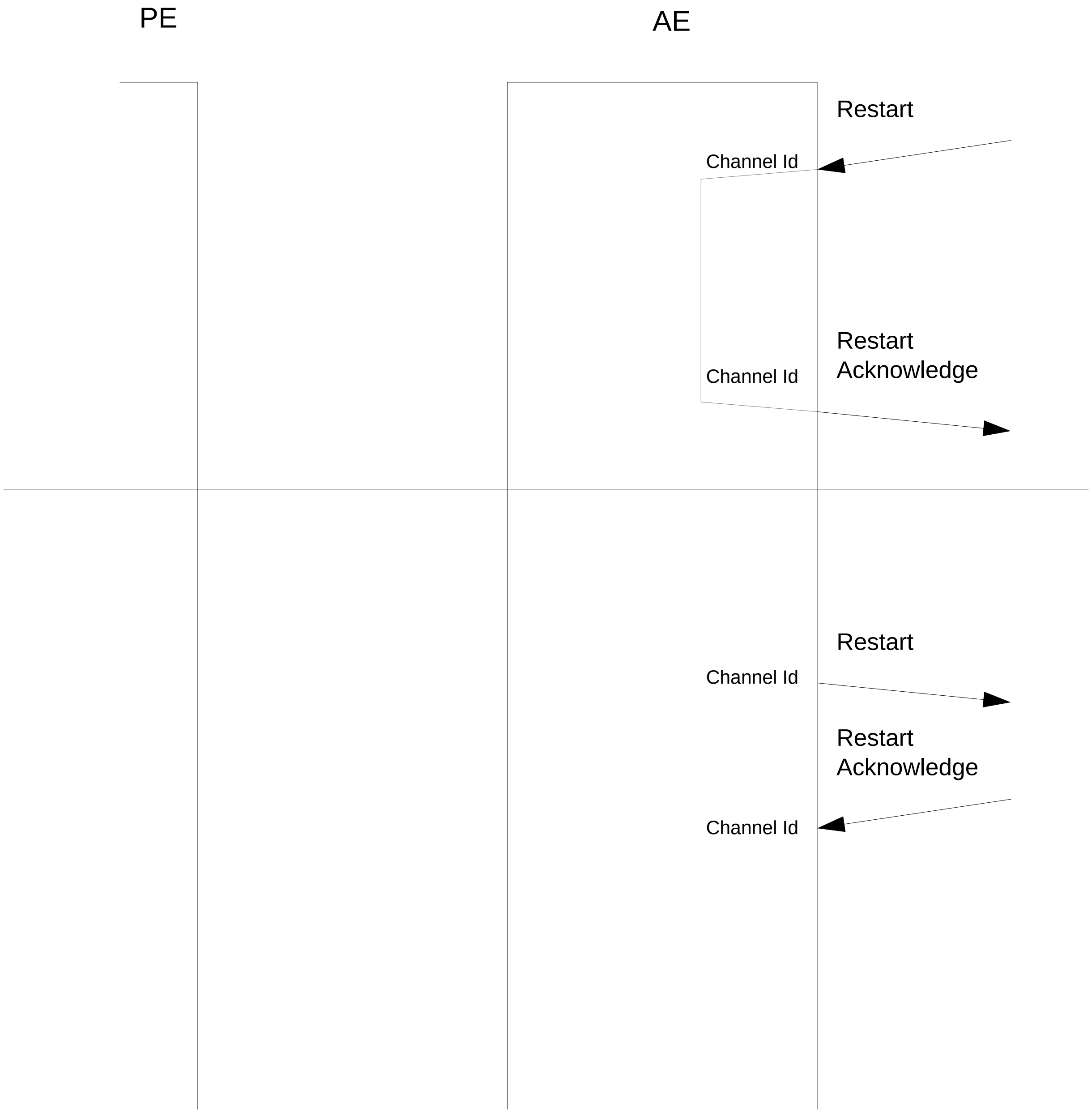


Figure A4.2
Other procedures.

Appendix B: References

B.1. ANS Documents

[1] ANSI T1.607-1990, *Telecommunications — Integrated Services Digital Network (ISDN) — Digital Subscriber Signalling System Number 1 (DSS1) — Layer 3 Signalling Specification for Circuit-Switched Bearer Service*.

B.2. CCITT Documents

[2] CCITT Recommendation I.320 - 1988, *ISDN Protocol Reference Model*.

[3] CCITT Recommendation I.515 - 1988, *Parameter Exchange for ISDN Networking*.

[4] CCITT Recommendation Q.921-1988 (also designated CCITT Recommendation I.441-1988), *ISDN User-Network Interface Data Link Layer Specification*.

[5] CCITT Recommendation Q.931-1988 (also designated CCITT Recommendation I.451-1988), *ISDN User-Network Interface — Layer 3 Specification for Basic Call Control*.

[6] CCITT Recommendation V.110 -1988, *Support of Data Terminal Equipments (DTEs) with V-series Type Interfaces by an Integrated Services Digital Network (ISDN)*.

[7] CCITT Recommendation V.120 -1988, *Support by an ISDN of Data Terminal Equipment with V-series Type Interfaces with Provision for Statistical Multiplexing*.

[8] CCITT Recommendation X.25 -1984, *Interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit*.

B.3. ISO Documents

[9] ISO 8824:1987(E), *Information processing systems — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*.

[10] ISO 8825:1987(E), *Information processing systems — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.

B.4. Other Documents

[11] NIST Special Publication 500-183, *Stable Implementation Agreements for Open Systems Interconnection Protocols*, Version 4, Edition 1, December 1990.

Appendix C: List of Acronyms

AE	ASI Entity
AEI	ASI Entity Identifier
ANS	American National Standard
AP	Additional Parameters
ASI	Application Software Interface
BRI	Basic Rate Interface
CCITT	English: International Telephone and Telegraph Consultative Committee
	French: Comité Consultatif International Télégraphique et Téléphonique
CPE	Customer Premise Equipment
CSD	Circuit Switched Data
CSV	Circuit Switched Voice
CUG	Closed User Group
DN	Directory Number
FIPS	Federal Information Processing Standard
ISDN	Integrated Services Digital Network
LAN	Local Area Network
LAPB	Link Access Procedure for the B-Channel
LAPD	Link Access Procedure for the D-Channel
LSB	Least Significant Bit
MSB	Most Significant Bit
NA	Network Adaptor
NSAP	Network Service Access Point
OSI	Open Systems Interconnection
PEI	Program Entity Identifier
POD	Packet On Demand
PCI	Programming Communications Interface for EuroISDN
PE	Program Entity
PRI	Primary Rate Interface
PSD	Packet Switched Data
SAPI	Service Access Point Identifier
SPID	Service Profile Identifier
TA	Terminal Adaptor
TEI	Terminal Endpoint Identifier